University of Nevada, Reno

Machine Learning based Mountainous Skyline Detection and Visual Geo-Localization

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in
Computer Science and Engineering

by

Touqeer Ahmad

Dr. George Bebis/Dissertation Advisor

December, 2017

THE GRADUATE SCHOOL

We recommend that the dissertation
prepared under our supervision by

TOUQEER AHMAD

Entitled

Machine Learning Based Mountainous Skyline Detection And Visual Geo-Localization

be accepted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

George Bebis, Ph.D., Advisor

Mircea Nicolescu, Ph.D., Committee Member

Bahram Parvin, Ph.D., Committee Member

Hung (Jim) La, Ph.D., Committee Member

Yantao Shen, Ph.D., Graduate School Representative

David W. Zeh, Ph. D., Dean, Graduate School

December, 2017

Abstract

With the ubiquitous availability of geo-tagged imagery and increased computational power, geo-localization has captured a lot of attention from researchers in computer vision and image retrieval communities. Significant progress has been made in urban environments with stable man-made structures and geo-referenced street imagery of frequently visited tourist attractions. However, geo-localization of natural/mountain scenes is more challenging due to changed vegetations, lighting, seasonal changes and lack of geo-tagged imagery. Conventional approaches for mountain/natural geo-localization mostly rely on mountain peaks and valley information, visible skylines and ridges etc. Skyline (boundary segmenting sky and non-sky regions) has been established to be a robust natural feature for mountainous images, which can be matched with the synthetic skylines generated from publicly available terrain maps such as Digital Elevation Models (DEMs). Skyline or visible horizon finds further applications in various other contexts e.g. smooth navigation of Unmanned Aerial Vehicles (UAVs)/Micro Aerial Vehicles (MAVs), port security, ship detection and outdoor robot/vehicle localization.

Prominent methods for skyline/horizon detection are based on non-realistic assumptions and rely on mere edge detection and/or linear line fitting using Hough transform. We investigate the use of supervised machine learning for skyline detection. Specifically we propose two novel machine learning based methods, one relying on edge detection and classification while other

solely based on classification. Given a query image, an edge or classification map is first built and converted into a multi-stage graph problem. Dynamic programming is then used to find a shortest path which conforms to the detected skyline in the given image. For the first method, we provide a detailed quantitative analysis for various texture features (Scale Invariant Feature Transform (SIFT), Local Binary Patterns (LBP), Histogram of Oriented Gradients (HOG) and their combinations) used to train a Support Vector Machine (SVM) classifier and different choices (binary edges, classified edge score, gradient score and their combinations) for the nodal costs for Dynamic Programming (DP). For the second method, we investigate the use of dense classification maps for horizon line detection. We use Support Vector Machines (SVMs) and Convolutional Neural Networks (CNNs) as our classifier choices and use normalized intensity patches as features. Both proposed formulations are compared with a prominent edge based method on two different data sets.

We propose a fusion strategy which boosts the performance of the edge-less approach using edge information. The fusion approach, which has been tested on an additional challenging data set, outperforms each of the two methods alone. Further, we demonstrate the capability of our formulations to detect absence of horizon boundary and detection of partial horizon lines. This could be of great value in applications where a confidence measure of the detection is necessary e.g. localization of planetary rovers/robots. In an extended work, we compare our edge-less skyline detection approach against deep learning networks recently proposed for semantic segmentation on an additional data set. Specifically, we compare our proposed fusion

formulation with Fully Convolutional Network (FCN), SegNet and another classical supervised learning based method.

We further propose a visual geo-localization pipeline based on evolutionary computing; where Particle Swarm Optimization (PSO) is adopted to find/refine an orientation estimate by minimizing the cost function based on horizon-ness probability of pixels. The dense classification score image resulting from our edge-less/fusion approach is used as a fitness measure to guide the particles toward best solution where the rendered horizon from DEM perfectly aligns with the actual horizon from the image without even requiring its explicit detection. The effectiveness of the proposed geo-localization pipeline is evaluated on a decent sized data set.

Dedication

Respectfully dedicated to,

My Father Professor (ret.) Muhammad Ikram

and

My Advisor Professor George Bebis, Ph.D.

Acknowledgements

First and foremost, I am very grateful to my advisor Professor George Bebis who has been a great inspiration, mentor, teacher, motivator, and facilitator throughout my doctoral degree. I would like to express my gratitude for his consistent guidance, encouragement, understanding, help and support on both technical and personal matters. I am very fortunate to have a supervisor like him, and will always be indebeted to him throughout my life. Secondly, I would like to thank Dr. Mircea Nicolescu, Dr. Bahram Parvin, Dr. Yantao Shen and Dr. Hung (Jim) La for serving in my dissertation committee and providing very useful comments and suggestions to improve the quality of my dissertation. I would also like to thank Dr. Martin Cadik from Brno University of Technology, Czech Republic, for his help with experiments in chapter six of my dissertation.

I would like to thank all my colleagues from Computer Vision Laboratory who joined for various timespans over the past five years and would specifically like to acknowledge company of Ebrahim, Deepak, Pourya and Mohammad. I extend my gratitude to the Department of Computer Science and Engineering, UNR for the financial support and providing me an opportunity to serve as a teaching assistant. I would like to thank Miss Lisa Cody

for all her help throughout these years. I would also like to acknowledge the financial support from National Aeronautics and Space Administration (NASA) and National Science Foundation (NSF) for funding various stages of this research.

I must thank my parents, my elder brother Touseef, my lovely wife Saba and my beautiful daughter Romaisa for always being on my side throughout this journey. Lastly, I would like to thank few of my friends; Omair, Wajahat, Haris, Abbas, Saadi, Mudasser and Haider for their support and help. Finally, I again thank my advisor Dr. George Bebis for all his guidance and supports in completing my Ph.D. dissertation.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Skyline Detection and Geo-Localization

In computer vision community, problem of segmenting an image into sky and non-sky regions is termed as horizon line detection or skyline extraction. Earlier methods for horizon line detection either rely on edge detection as preprocessing step [3, 21, 28, 29] or address the problem using supervised (classification) [5, 6, 7, 9, 19] or unsupervised (clustering) [4] machine learning techniques. Edge based methods suffer due to edge ambiguities and result into gaps due to clouds thus making part(s) of horizon occluded and edge detector may miss it. On the other hand, non-horizon edges might be included into solution horizon due to bias induced towards specific solutions [3]. Machine learning based attempts mostly model sky and non-sky regions and are based on the faulty assumption that horizon is a linear boundary [6, 19]. Moreover, the non-sky regions can vary a lot and it is very hard for machine learning algorithms to generalize across all these variations in

the test sets [4, 8]. For example, a non-sky region could be comprised of water, mountains, plains or mixture of all i.e. significant color and texture variations.

In addition of being useful for applications such as port security and ship detection[19, 20, 60, 61], augmented reality [31], UAV/MAV stability and navigation [4, 5, 6, 7, 8, 9, 54, 55, 56], and planetary and outdoor robot localization [10, 11, 12, 57, 58, 59]; horizon/skyline has proven to be useful for visual geo-localization of mountainous imagery[17, 36, 28, 29]. The geolocation of natural scenes is more challenging compared to urban environments due to lack of stable man-made structures and geo-referenced street imagery. The urban visual geo-localization of frequently visited tourist attractions has been the true focus of research due to easy availability of geo-tagged imagery. Recently, some attempts have been made towards geo-localization of natural/mountain scenes. In addition to the lack of geo-tagged data, natural visual geo-localization also suffers from changed vegetation, lighting and seasonal changes. Typical approaches for mountain/natural geo-localization rely on mountain peaks and valley information, visible skylines, ridges or combinations of all three [17, 28, 29, 31, 32, 33]. Skyline has been established to be a robust natural feature for mountainous images which can be matched with the synthetic skylines generated from publicly available terrain maps e.g. Digital Elevation Models (DEMs). Hence, the very first step in the geolocation pipeline for mountainous regions is to find the skyline in the given query image. Most of the solutions for mountainous geo-localization rely on **user-in-the-loop** methods for skyline extraction where a user is required to mark/correct portion of the skyline [17, 29, 32, 33].

1.2 Contributions

In this work, we have proposed two novel machine learning based horizon line detection approaches. Instead of modeling sky and non-sky regions we model horizon and non-horizon regions. It is important to note that horizon boundary is more consistent as compared to non-sky regions, therefore methods trained on horizon are better as compared to others trained on sky, non-sky regions. Moreover, earlier methods are based on faulty assumptions e.g. horizon being a linear boundary [6, 19] or horizon being present in the upper half of the image [3] which is generally not the case. Our first proposed method relies on edge detection and classification whereas the second addresses the problem in a pure classification framework. Both approaches formulate the resultant edge/classification map as a multi-stage graph problem and find a shortest path using Dynamic Programming (DP) which conforms to the detected horizon in the given query image.

For the first approach, given a query image; Canny edge detector is applied first with a range of thresholds to keep only those edges which survive over a wide range. The surviving edges are termed as Maximally Stable Extremal Edges (MSEEs). The number of edges is further reduced significantly by classifying the MSEEs into horizon and non-horizon edges using a trained classifier. Dynamic programming is then applied on horizon classified edges using any of the available nodal costs. We investigate the suitability of various local texture features and their combinations as feature choices for the trained horizon classifier. Specifically, we explore SIFT [14], LBP[13], HOG[15] and their combinations SIFT-LBP, SIFT-HOG, LBP-HOG and SIFT-LBP-HOG

as features to train an SVM [26] classifier. We evaluate various nodal costs for dynamic programming e.g. binary edge scores, normalized classification scores for horizon classified edges, gradient information and their combination.

In the second proposed approach, we use machine learning and Dynamic Programming to extract the horizon line from a classification map instead of an edge map. The key idea is assigning a classification score to each pixel, which can be interpreted as the likelihood of the pixel belonging to the horizon line, and representing the classification map as a multi-stage graph. Using DP, the horizon line can be extracted by finding the path that maximizes the sum of classification scores. In contrast to conventionally used edge maps which are typically binary (edge vs no-edge) and contain gaps, classification maps are continuous and contain no gaps which yields significantly better solutions. We use normalized intensity patches as feature choice for this method and train SVM [26] and CNN [27] classifiers.

Both of our formulations allow us to remove certain assumptions which are common to earlier methods such as the horizon is close to the top of the image or that the horizon forms a straight line. The purpose of these assumptions is to either bias the DP solution or simplify the underlying segmentation problem but they fail to produce good results when not valid. The proposed formulations are compared with an earlier prominent approach which relies only on edge information and is based on faulty assumption of horizon being present in the upper half of the image [3]. The results are reported for two challenging data sets (125 images in total) i.e. Baslat Hills data set (45 images) and Web data set (80 images) comprised of mountainous images

captured during an outdoor robot exploration and randomly chosen from the web with considerable viewpoint, scenic and weather changes. For both approaches although our training set is comprised of a very small number of images from the same location, yet our results illustrate that our methods generalize well to images acquired under different conditions and geographical locations and outperform the traditional method based only on edges.

Using the same data sets, we evaluate the performance of each of our proposed approaches and identify specific cases where one outperforms the other. Next, we propose a straight forward fusion strategy which boosts the performance of the edge-less approach using edge information. The fusion approach, which has been tested on an additional data set [17] – CH1 data set; outperforms each of the two proposed methods alone.

Our proposed framework allows verification of a found horizon solution and is capable to detect absence of horizon boundary and detection of partial horizon lines. In many applications where the horizon line is used for rover/robot localization and navigation, it is important not only to detect the horizon line but also to report a confidence measure of the detection. This is useful in many cases, for example, when the horizon line in not visible in the image. A by-product of the proposed edge-less/fusion approaches is a confidence measure corresponding to the normalized sum of classification scores along the path found by DP. Using a Bayesian approach, we can determine whether the path found corresponds to the horizon line or some other irrelevant path. This could also be used as a validation step where the method misses the actual horizon line and finds another solution instead. The statistical measure along the found path can be used to reject such faulty

detections. Moreover, we demonstrate how the proposed approach can be adapted to handle partially visible horizon lines. This is quite useful since localization is still feasible, at least from a theoretical point of view, using partially visible horizon lines. We demonstrate both capabilities of our proposed formulation on additional images which are (i) either missing a skyline completely or (ii) have partial skylines.

Next, we provide the comparison of our edge-less horizon detection approach with three other recently proposed methods; one focused on visual geo-localization but relying on accurate skyline segmentation [33] and other two methods proposed for general semantic segmentation – Fully Convolutional Networks (FCN) [42] and SegNet[43]. Each of the classical machine learning based methods is trained on a common training set [17] (CH1) which comprises of 203 images whereas models for the deep learning methods are fine tuned for sky segmentation problem through transfer learning using the same data set. Each of these four methods is tested on an extensive test set (about 3K images) covering various challenging geographical, weather, illumination and seasonal conditions from Alps. We report mean dice coefficient and average absolute pixel error for each of the presented formulation.

Using our edge-less/fusion framework, we propose a mountainous visual geo-localization pipeline which relies on evolutionary computing specifically Particle Swarm Optimization (PSO). Given previous rough estimates of the camera orientation and known Global Positioning System (GPS) and camera intrinsics, PSO tries to find the orientation estimates that perfectly align the rendered horizon from a 3D Digital Elevation Model with the one viewed in the 2D imagery. We use the dense classification score image resulting

from our horizon classifier as the fitness measure for PSO. For a given PSO particle (orientation estimate), the synthetic horizon is rendered from DEM and overlaid on the classification score image. The horizon-ness score of all the pixels belonging to rendered horizon is average out which serves as the fitness of the current solution (orientation estimate). This idea stems from the observation that pixels belonging to the vicinity of true horizon have more classifications scores. Hence the particles would drive towards this narrow search space around true horizon where a best solution is eventually found that aligns well with the true horizon.

Our final contribution is the evaluation metrics being used to measure the performance of horizon line detector. It should be noted that all performance comparisons reported in this thesis are based on the absolute average error between the solution found and ground truth. Previously, detection results were reported in the form of a percentage (e.g., what percentage of the horizon line was detected) which does not provide a true quantitative evaluation of horizon line detection methods. To the best of our knowledge, the only exception is the work of Hung et al. [2] who also reported the absolute average error.

1.3 Thesis Organization

Following this chapter, we provide a brief literature review (chapter 2) of earlier horizon detection methods and visual geo-localization approaches and list the details of a prominent horizon line detection method by Lie et al.[3] which relies solely on edge detection. In chapter 3 we outline the details

of our proposed edge-less and edge-based horizon line detection methods. Chapter 4 provides the experimental details and results for different data sets for both proposed methods and their comparisons against Lie et al. [3] formulation. Additionally, this chapter details the proposed fusion approach which is evaluated on an additional data set. In chapter 5, we demonstrate the capability of proposed framework to detect the absence of horizon and partial horizon detection. Chapter 6 presents the comparison of our edge-less approach against three other methods on an extensive data set. In chapter 7, we describe our proposed geo-localization pipeline and detail some results. Thesis is concluded in chapter 8 with concluding remarks about the presented methods and listings of the future work which can be further explored in follow up studies.

Chapter 2

Literature Review

2.1 Skyline Detection

Horizon/Sky line detection or sky segmentation is the problem of finding a boundary between sky and non-sky regions (ground, water or mountains) given a gray scale or color image. Previous attempts to horizon line detection can be categroized into two major groups; methods modeling sky and non-sky regions either by some machine learning algorithms – both supervised and unsupervised approaches [4, 5, 6, 7, 9, 19] or methods relying on edge detection as the essential pre-processing step [3, 21, 28, 29]. Recently some attempts [1, 2] have been made to combine these two ideas to refine the edges by training classifiers but broadly these attempts also fall under the second category as they are not possible without edges being detected. Most of the earlier methods to horizon detection suffer from the assumption of horizon boundary being linear and hence are limited.

In [6], Ettinger et al. proposed the use of horizon line for flight stabil-

ity and control of MAVs. Their horizon detection approach is based on the assumptions that the horizon is linear and it segments the image into two regions of significantly different appearance (sky and non-sky). Using RGB color information, the sky and ground regions are modeled using Gaussian distributions. Since it was assumed that the sky and ground regions follow a Gaussian distribution, which is not always valid, Todorovic et al. [9] proposed a general statistical image modeling framework to build prior models for sky and ground. Unlike the work in [6], they found both color and texture to be critical for building priors. They used color (Hue, Intensity) and texture (Complex Wavelet Transform (CWT), magnitude only) to train a Hidden Markov Tree (HMT) model using the expectation maximization (EM) algorithm. The posterior likelihoods for two classes at different scales are fused together and Bayesian segmentation is performed to separate the sky and non-sky regions. McGee et al. [7] used sky segmentation as an obstacle detection tool for small scale UAVs. They trained an SVM classifier based on YCbCr color information to classify pixels into sky and non-sky regions. Morphological erosion and dilation operators were applied on the resultant binary image to rectify misclassifications. Next, they used Hough Transform on the border pixels to cast votes for candidate linear horizon boundaries. The approach of Fefilatyev et al. [19] is also based on the horizon boundary being linear; it uses color and texture features such as mean intensity, entropy, smoothness, uniformity etc. to train an SVM, a J48 and a naive Bayes classifier. Their experiments are limited to two sets of ten images each and their method fails to detect good linear horizons for two out of ten images due to reflection of water and presence of fog. In [36], Liu et al.

have proposed a sensor fusion approach to estimate the horizon line using a textured Digital Elevation Map (DEM), an airport model, GPS, AHRS and vision sensors. Their objective was to estimate an accurate linear horizon boundary from an aircraft in low visibility conditions; their approach does not generalize to non-linear horizons.

Croon et al. [5] extended the features used in[19, 6, 9, 7] by including corner-ness, grayness and Fisher Discriminant features to train shallow decision trees. Their approach was tested in the context of MAVs for obstacle avoidance and is able to detect non-linear horizon boundaries. The fusion-based approach of Yazdanpanah et al.[22, 62] combines the output of a Neural Network (NN) with K-means clustering and is based on the same texture features such as in [5, 19]. Their system is based on various heuristics and parameters setting that might not generalize well to different data sets. In [4], Boroujeni et al. also rely on clustering for horizon line detection. Their method is based on the assumption that a dominant light field exists between sky and non-sky regions right above the horizon. They have investigated K-means and intensity based clustering to find this light field in various images. In general, the assumption about the presence of a light field is not always valid under different seasonal conditions or geographical locations and the data set being used in [4] is not general enough to justify such a strong assumption. Thurrowgood et al. [8] used horizon detection for UAV attitude estimation. In their approach, a projection onto a single line in the RGB color space is found by minimizing the overlap of the sky and non-sky classes. This is somewhat similar to the Fisher Discriminant used by Ettinger et al. in [6] but it is computationally less expensive. In

[35], Neto et al. proposed a robust horizon line detection algorithm using Otsu segmentation and Hough transform for real-time autonomous navigation. Braun and Singhof [41] proposed a seed growing algorithm where the top row of the image is assumed to be part of the sky. This sky region is then grown based on the assumption that the distance of the current pixel's brightness from a local brightness mean is smaller than a fixed percentage of the global standard deviation. This method is heavily dependent on the percentage factor. Moreover, the results reported are based on a very small image set (18 images) from the Switzerland data set [17] and no quantitative comparison with the ground truth is provided.

Dusha et al. [39] used horizon line information with optical flow for attitude estimation of fixed-wing aircrafts. In their approach, edge detection is performed separately in each smoothed color channel. The detected edges from each channel were combined in a single map and the horizon line was found using Hough voting. Shen et al. [38] proposed an edge-based hierarchical approach for horizon line detection where coarse-level detection was performed first followed by fine-level adjustments. They successively performed Canny edge detection and Hough voting on a low pass filtered image to find the strongest lines. Using the five highest peaks in the Hough space, the best line was chosen based on average edge strength. The straight line chosen is refined, essentially becoming non-linear, using edge position and strength information. The results reported are based on synthetic images generated from Google maps which do not reflect if the idea generalizes to real images. Gershikov [40] provides a comparison between gray scale and color based horizon detection methods which rely on edge detection and

Hough transform. In [37], Shen and Wang proposed a simple gradient magnitude based sky segmentation approach. By redefining the energy function proposed by Ettinger et al. [6], their approach is applicable to general curves instead of a linear boundary.

The most prominent method belonging to the edge detection based approaches is that of Lie et al. [3] where horizon line detection is formulated as a graph search problem. Their approach relies on edge detection and assumes a consistent edge boundary between sky and non-sky regions. The detected edge map is represented as a multi-stage graph where each column of the image becomes a stage in the graph and each edge pixel becomes a vertex. The shortest path, extending from the left-most column to the right-most column, is then found using DP. It should be mentioned that the assumption that the horizon boundary is a consistent edge boundary is rarely true in practice due to environmental conditions (e.g., clouds, fog, mist) and edge gaps. To address the issue of gaps, Lie et al. [3] have proposed a gap-filling step which highly depends on the choice of certain parameters. Moreover, they assume that the edges in the upper half of the image belong to the horizon boundary and hence introduce a bias to find the horizon solution in that region. However, the edges in this region may very well be due to the presence of clouds.

2.1.1 Lie et al. [3]

This section briefly details the approach used by [3]. Since, we also formulate our approaches as a multi-stage graph problem, it seems necessary to

provide the details of original approach by [3] so that we can point out the problems with underlying assumptions by [3] and identify various similarities and differences of our proposed approaches.

Lie et al. [3] formulated the problem of horizon detection as a multi-stage graph search problem. Given an image, edge detection is applied first. The detected edges are then used to form a multi-stage graph and DP is applied to extract the horizon line by finding the shortest path. This approach is based on the assumption that the horizon is present in the upper half of the image. Specifically, given an image of size $M \times N$, edge detection is performed to compute a binary edge map E where 1 implies the presence of an edge pixel and 0 a non-edge pixel.

$$E(i,j) = \begin{cases} 1, & \text{if } (i,j) \text{ is an edge pixel.} \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

The edge map is used to build an $M \times N$ multi-stage graph $G(V, E, \Psi, \Phi)$ where each pixel in the edge map corresponds to a graph vertex; a low cost l is associated with edge pixels while a very high cost (i.e., ∞) is associated with non-edge pixels as shown below:

$$\Psi(i,j) = \begin{cases} l, & \text{if } E(i,j) = 1. \\ \infty, & \text{otherwise.} \end{cases} \tag{2.2}$$

$\Psi(i,j)$ is the cost associated with vertex i in stage j (i.e., v_{ij}). The graph can be visualized as an N (columns) stage graph where each stage contains M nodes (rows). To deal with edge gaps, a gap filling approach was proposed.

Given a node i in stage j, its neighborhood in the next stage $j+1$ is defined by a δ parameter, that is, the number of nodes to which i could be connected in stage $j+1$. The edges from i to its neighbors are associated with costs equal to the vertical absolute distance from it as shown in the equation below.

$$\Phi(i,k,j) = \begin{cases} |i-k|, & \text{if } E(i,j) = E(k,j+1) = 1 \\ & \text{and } |i-k| \leq \delta \\ \infty, & \text{otherwise.} \end{cases} \tag{2.3}$$

If a node i in stage j cannot be connected to any node in stage $j+1$ within δ neighborhood, a search window is defined using δ and tolerance-of-gap (tog). Once an edge node k is found in the search window, the gap filling is performed by introducing dummy nodes between node i in stage j and node k. A high cost is associated with the dummy nodes introduced by the gap filling step.

Once the gaps have been filled with high cost dummy nodes, the cost of the nodes in stages 1 and N is increased based on the vertical position of the nodes as shown by the equation below:

$$\Psi(i,j) = \begin{cases} (i+1)^2, & \text{if } j = 1 \text{ or } j = N \\ \Psi(i,j), & \text{otherwise.} \end{cases} \tag{2.4}$$

This enforces the assumption that the horizon line is present in the upper half of the image and hence biasing the DP solution towards a shortest path present in the upper half. Next, a source s and a sink t are added to the left of the left most stage (i.e., stage 1) and to the right of the right most

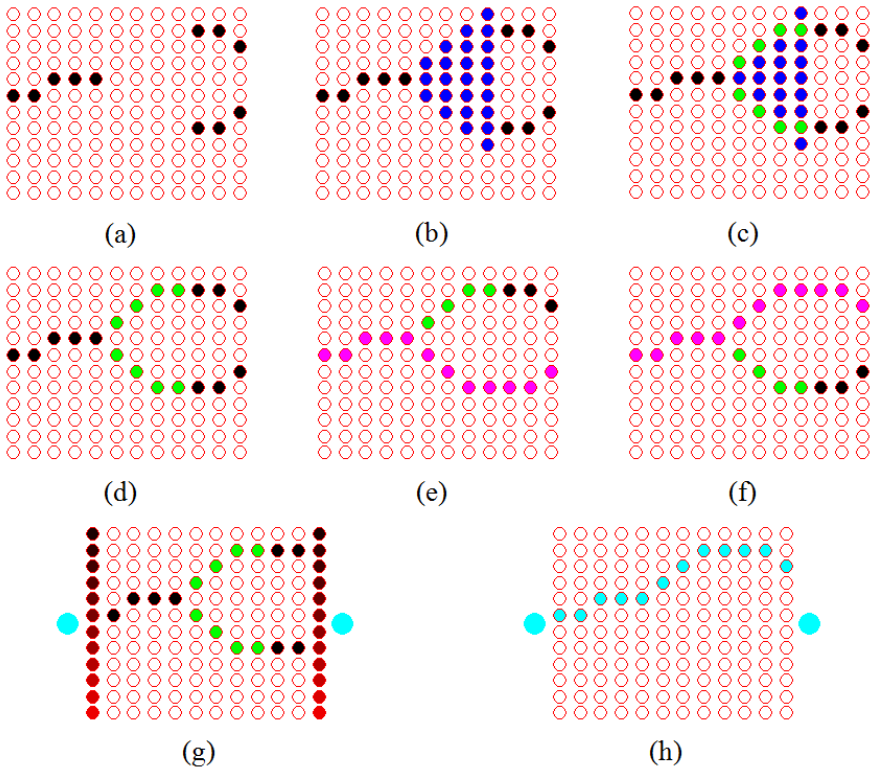

Figure 2.1: Illustration of the horizon detection steps using the method of Lie et al. [3]

stage (i.e., stage N) respectively. A zero cost is associated with each one of them. The s node is connected with all the nodes in stage 1 and all the nodes in stage N are connected to node t. The horizon boundary is detected by finding the shortest path extending from node s to t using DP.

Figure 2.1 illustrates the steps of Lie et al. [3] for a sample image. An edge map is shown in Figure 2.1-(a) where black and white circles represent edge and non-edge pixels respectively. A search window (highlighted by blue circles) is shown in Figure 2.1-(b) for the edge node in stage $j = 5$ using $\delta = 1$ and $tog = 4$. Within the search window j+tog, two edge nodes are

discovered which are then connected to node j by introducing dummy nodes as shown in Figure 2.1-(c,d) (highlighted in green). So, there exist two equal cost paths 2.1-(e,f) in the resultant image, highlighted in magenta. However, the nodes in stage 1 and N are set to a higher cost associated with their vertical positions; this is reflected by an increasing intensity in Figure 2.1-(g). Two nodes s and t (cyan) are then introduced, as described above, and DP is applied on this graph. As shown in figure 2.1-(e,f) the two paths have the same cost, however the bias introduced in 2.1-(g) would make the upper path of lower cost and DP will select this path due to the assumption of the horizon line being present in the upper half. However, it might be possible that the true horizon line is actually the lower one and that the upper edge segment was only due to some clouds.

We would show various specific examples in another chapter where this method failed to detect a portion of horizon due to this bias associated with the location of horizon.

2.2 Visual Geo-Localization

With the massive availability of geo-tagged imagery and increased computational power, geo-localization or geolocation has captured a lot of attention from researchers in computer vision and image retrieval communities. Significant progress has been made in urban environments with stable man-made structures and geo-referenced street imagery of frequently visited tourist attractions [44, 45, 46]. Recently some attempts have been made towards geo-localization of natural/mountain scenes which is more challenging due

to changed vegetations, lighting and seasonal changes and lack of geo-tagged imagery. Typical approaches for mountain/natural geo-localization rely on mountain peaks and valley information, visible skylines, ridges or combinations of all three [63, 64, 28, 17, 29, 31, 66, 32, 65, 33]. Skyline has been established to be a robust natural feature for mountainous images which can be matched with the synthetic skylines generated from publicly available terrain maps e.g. Digital Elevation Models (DEMs). Hence, the very first step in the geolocation pipeline for mountainous regions is to find the skyline in the given query image. Most of the solutions for mountainous geo-localization rely on **user-in-the-loop** methods for skyline extraction where a user is required to mark/correct portion of the skyline [17, 29, 32, 33]. Making a truly autonomous horizon/skyline detector would definitely advance this research dimension.

Using silhouette edge matching, Baboud et al. [28] estimate the pose of camera relative to geometric terrain model assuming known viewpoint and FOV estimates. Effectively, a rotation $g \in SO(3)$ is searched which maps the camera frame to the terrain frame. They developed a robust silhouette matching metric to cope with inevitable noise affecting detected edges (compass edge detector is used). Since, a direct extensive search on $SO(3)$ is based on their devised metric is quite expensive, therefore they also proposed a preprocessing search space reduction step based on spherical cross-correlation of 2D edge orientation vectors. They reported that 86% of 28 images were correctly aligned belonging to two distinct mountain regions with matching error below 0.2°. Baatz et al. [17] proposed a visual geo-localization pipeline based on bag-of-curvelets; where shape information is aggregated across the whole

skyline of a query image and a similar configuration of shapes is searched in a large scale database of panoramic skylines (extracted offline from DEMs). In addition to encoded contourlets, the viewing direction for each descriptor is also saved which is used for on-the-fly geometric verification in an inverted file search framework. Since, they are comparing $10^\circ - 70^\circ$ views with 360° panoramas, they redefine the weighted L1-norm to implement "contains"-semantics instead of conventional "equal"-semantics used for visual words (curvelets) matching. The most promising coarse estimation for the viewing azimuth direction is used to initialize Iterative Closest point (ICP) while keeping other two angles at zero which determines a full 3D rotation. The average alignment error between two visible horizons is used to re-rank the candidates in ICP framework. They reported an 88% recognition rate on their challenging data set comprised of more than 200 images where determined position was within one kilometer radius of the ground truth. It should be noted that about half of their images required **manual interaction** at the sky segmentation stage.

Similar to Baatz et al. pipeline [17], Tzeng et al. [29] also proposed a localization approach which they used specifically for desert imagery. However, instead of curvelet features, concavity-based features across query and synthetic skylines are used for matching without any use of meta-data such as GPS, FOV and focal length. In contrast to [17] where overlapping curvelet descriptors are generated for pre-defined angular width, they generate concavity descriptors around detected points of extreme curvature. Furthermore, a similarity transformation is applied on features to achieve scale and in-plane rotation invariance. The endpoint matching and features shape matching is

accomplished through geometric hashing and k-d trees respectively. Ranked database skylines from both matchers (endpoint and shape) are further refined using alignment error based on sampling of overlapping regions between query and database skylines. It is interesting to note that in their method, skyline in the query image is first roughly **marked by a user** and further refined by edge detection and Dynamic Programming framework as detailed in Lie et al. [3].

Porzi et al. [31] also addressed the same image-to-world registration problem, however, in the context of an Augmented Reality based smart phone application. They first computed rough estimates for position and orientation from phone's on-board GPS and inertial sensors. These estimates are then refined by matching the skylines extracted from images taken by phone's camera and rendered from DEMs generated on a server. In principle, their approach is closer to that of [28] since they also assume roughly known position and orientation. However, they rely on a learning based edge filtering approach which results in an improved accuracy and computational cost desirable for a smart phone application. Based on the orientation estimation from device's inertial sensors, skyline detected from the phone's camera image and rendered profiles received from the server define a search space around the rough estimate which is explored by Particle Swarm Optimization (PSO) for refined orientation estimation [31]. This is accomplished by maximizing the objective function based on the matching between the skyline contour and contours projected (pin-hole camera projection model) from profiles received from the server. Porzi et al. extended their work in [69], where they explored various formulations of the original approach [31]

along with different edge detectors and objective functions.

Like Porzi et al. [31], the work of [67] is also focused on an augmented reality application; where extracted skyline is matched with renderred skyline in order to provide the local geo-spatial information to the mobile user. Inspired from [68], Nicolle et al. [67] also generate classification score map for the query skyline, however they rely on only pixel differences in RGB color space due to real-time requirement of their application. To align the renderred skyline from DEM with the classification score map, first a search space is defined based on camera FOV and then a 3D search is conducted in an iterative fashion. They encode the renderred skyline as set of 2D vectors and use the weighted classification score map as their energy function. To explore the full search space, at each iteration more vectorized DEM points are added and transformations resulting in lower scores based on energy function are definitively removed from search space. This two step exploration continues until the vectorized skyline is tested at full resolution. Nicolle et al. [67] reported an average vertical distance of 3 pixels between the ground truth and DEM based aligned skylines for their test set of 20 images.

Chapter 3

Proposed Skyline Detection Methods

This chapter outlines the technical details of our two proposed skyline detection methods. In section 3.1, we briefely describe the steps involved in the training process for the first method i.e. the edge-based method. Section 3.2 provides the details of detecting a skyline in a query image using this method. Section 3.3 lists the details of different nodal costs which we have explored for our first proposed method. Sections 3.4 and 3.5 respectively provide the details of training and testing phase of our second proposed method.

3.1 Edge-based Method

In the first approach, we propose a machine learning based framework where number of edges are reduced considerably first by Maximally Stable Extremal Edges (MSEEs) and then by using a trained SVM classifier. Different

components of this method are presented next.

3.1.1 Maximally Stable Extremal Edges (MSEEs)

The idea of extracting MSEEs is inspired from extracting Maximally Stable Extremal Regions (MSER) [25]. Given a gray scale image, we compute the edge image using the Canny edge detector [30] with sigma (σ) parameter being fixed value while varying the low and high thresholds. This results in the generation of N binary images assuming N combinations of parameter values have been used. The resultant binary images are named as E_1 to E_N. An edge at pixel location (x, y) is considered stable if it is detected as an edge pixel for k consecutive threshold values. The image comprised of these stable edges is referred as MSEE image and is denoted by E_m. Mathematically,

$$E_m(i,j) = \begin{cases} 1, & \text{if } \sum_{n=1}^{N} E(i,j)_n > k. \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

where, E_n is a binary edge image corresponding to nth combination of Canny parameters. In our experiments, we varied the high threshold of the Canny edge detector, Th, between 0.05 and 0.95 with a step of 0.05; the lower threshold Tl was set to be $0.4 \times Th$. It was found through experimentation that $\sigma = 2$ and $k = 3$ were optimal choices. The computation of MSEE Image reduces the number of edges considerably while keeping the important horizon edges. Figure 3.1 shows a sample image, the output of the Canny edge detector and the MSEEs. It can be observed that the number of edges have remarkably been reduced in MSEE image while the edges belonging to

the horizon line are maintained.

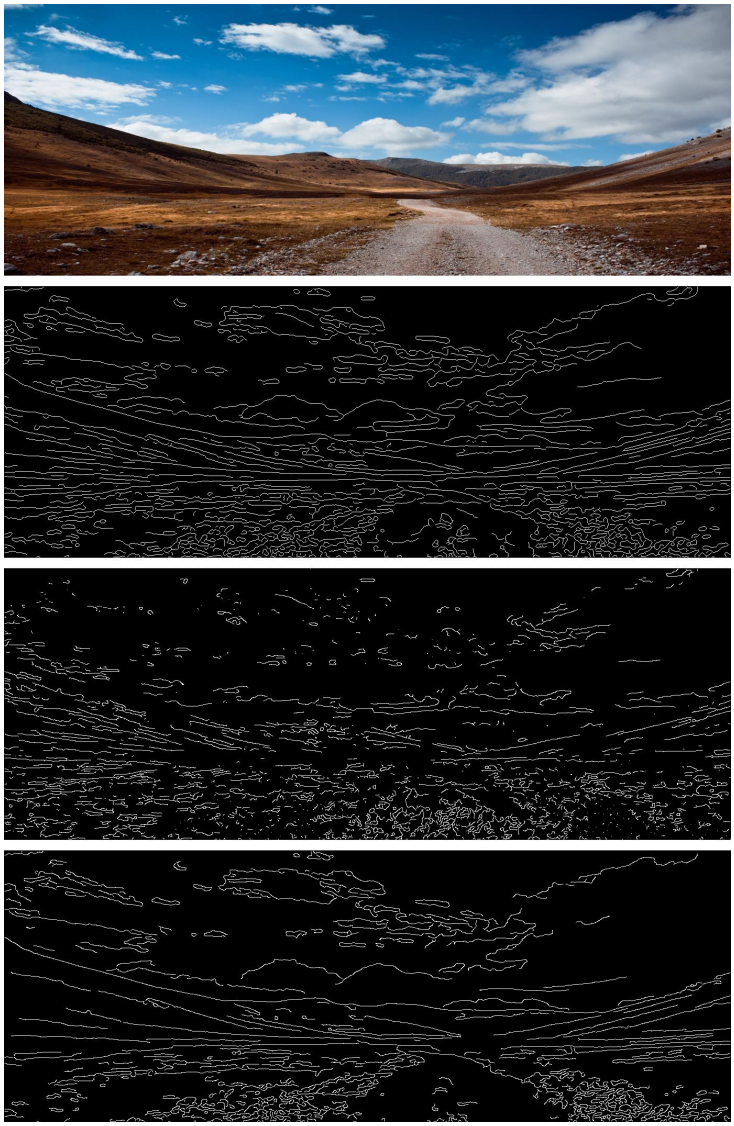

Figure 3.1: Effect of MSEE: (row1) input image, (row2) output of the Canny edge detector, (row3) discarded edges by MSEE and (row4) survived edges i.e. MSEE image.

3.1.2 Key Points Selection

To train the SVM classifier, we manually label the horizon line pixels in the training images using the corresponding MSEE images. A careful manual labeling of horizon pixels is required because : (i) some portion of the true horizon line might not be detected as edges and/or (ii) the edge detector's output might not match the true horizon line perfectly and/or (iii) there may be edges which are not horizon pixels but strong enough to survive different parameter choices. In our experiments, these manually labeled horizon pixel indices are used for comparing the detected horizons with the ground truths in our experiments. Figure 3.2 shows few images from Web data set with ground horizons marked as red.

Figure 3.2: Ground truth horizon lines; highlighted in red.

The positive and negative key points for all the images in the training set are chosen uniformly and randomly from the ground truth horizon location and MSEE non-horizon edge locations respectively. Figure 3.3 shows the locations of positive and negative key points for one of the training image from Basalt Hills data set.

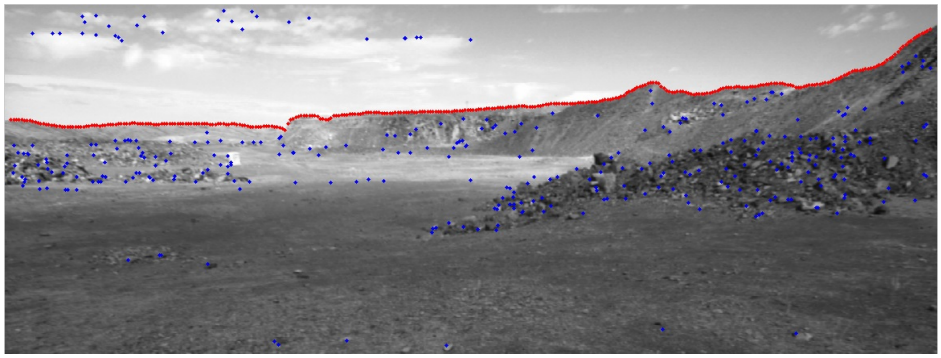

Figure 3.3: Positive (red) and negative (blue) key point locations for one of the training image.

3.1.3 Features and Classifier Training

To train an SVM classifier, we take a 16×16 window around each positive and negative key point and compute the chosen descriptor. We have investigated three texture descriptors as feature choices to train the SVM classifier namely Scale Invariant Feature Transform (SIFT) [14], Local Binary Patterns (LBP)[13] and Histogram of Oriented Gradients (HOG)[15]. We also explore their combinations with each other and then all of them combined as feature choices for our classifier.

We use the implementation of these descriptor available from *vlfeat toolbox* [16] which provides 128, 58 and 31 dimensional vectors for SIFT, LBP and HOG respectively. For each descriptor, an SVM classifier is trained using the positive and negative instances from the training set. The combinations of the descriptors are formed by mere concatenation of the feature vectors for each training and testing instance. The feature sizes for each of the combinations : SIFT-LBP, SIFT-HOG, LBP-HOG and SIFT-LBP-HOG are 186, 159, 89 and 217 respectively. We have found SIFT-HOG combina-

tion as the best choice when compared with individual descriptors and other combinations as described in the next chapter.

Figure 3.4 shows a flow diagram for the training phase of our edge-based skyline detection approach.

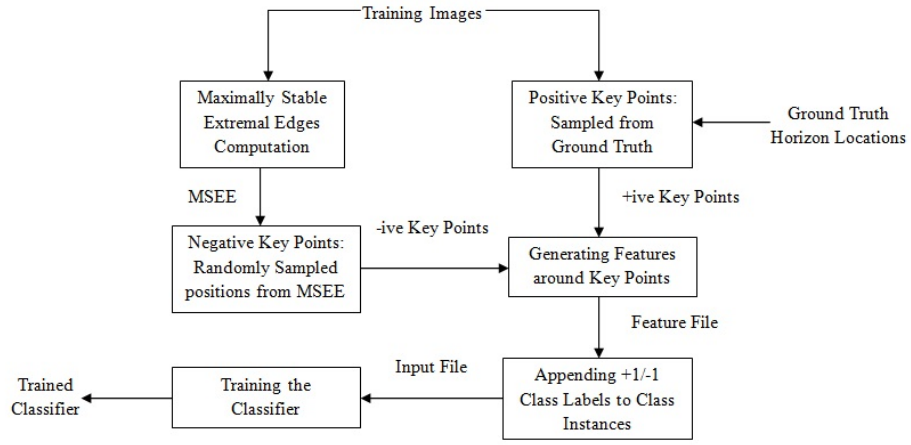

Figure 3.4: Flow diagram for the training phase of the proposed edge-based approach.

3.2 Skyline Detection

This section describes various steps involved for the detection of horizon line in a given query image. Figure 3.5 shows various steps of the test phase of our edge-based approach.

3.2.1 Filtering MSEE Pixels

In the test pipeline, MSEE image E_m is generated for a given query image according to equation 3.1 . Each of the edge pixels in MSEE image is treated

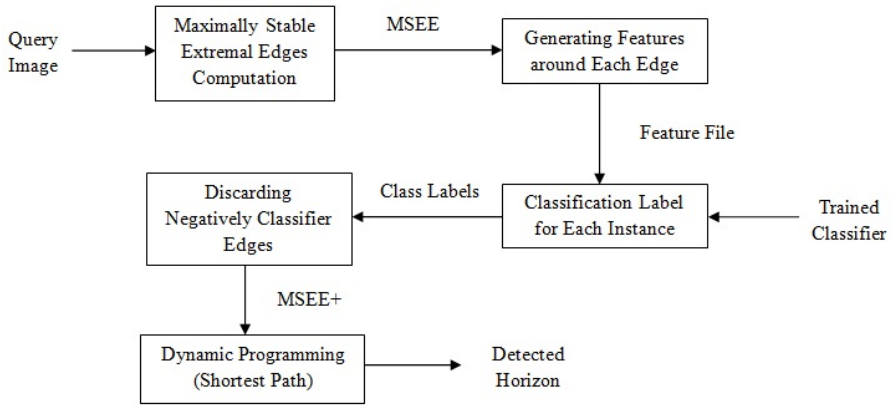

Figure 3.5: Flow diagram for the testing phase of the edge-based approach.

as a key point. The chosen descriptor is computed around the key point and then classified as horizon or non-horizon using the trained classifier. The resultant edge image comprises of only horizon classified edges is named E_+. If the classifier is assumed to be a binary function assigning 1/0 labels to the input edge pixel as described in eq. 3.2,

$$C(i,j) = \begin{cases} 1, & \text{if } (i,j) \text{ pixel is classified as horizon.} \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

then mathematically E_+ can be written as,

$$E_+(i,j) = \begin{cases} 1, & \text{if } E_m(i,j) = 1 \text{ and } C(i,j) = 1. \\ 0, & \text{otherwise.} \end{cases} \tag{3.3}$$

In addition to the reduction caused by MSEE as demonstrated in figure 3.1; horizon pixel classification further significantly reduces the number of horizon candidate edges. Figure 3.6 shows an example to highlight the sig-

nificant reduction in number of horizon candidate edges for a query image.

Figure 3.6: Effect of the classifier: (row1) A sample image, (row2) corresponding MSEE and (row3) MSEE_+ images. Note the reduction in number of edges due to the classifier.

3.2.2 Graph Formulation for Dynamic Programming

Instead of applying the Dynamic Programming directly on the output of the edge detector, we use the horizon classified edge image E_+ to formulate the multi-stage graph $G(V, E, \Psi, \Phi)$. So, the equations 2.2 and 2.3 are modified

accordingly.

$$\Psi(i,j) = \begin{cases} l, & \text{if } E_+(i,j) = 1. \\ \infty, & \text{otherwise.} \end{cases} \tag{3.4}$$

$$\Phi(i,k,j) = \begin{cases} |i-k|, & \text{if } E_+(i,j) = E_+(k,j+1) = 1 \\ & \text{and } |i-k| \leq \delta \\ \infty, & \text{otherwise.} \end{cases} \tag{3.5}$$

Since, the number of candidate horizon edges are reduced considerably due to the use of MSEE and the trained classifier therefore we do not enforce the bias towards horizon solutions in the upper half. So the equivalent to equation 2.4 is not required in our formulation. However, any kind of gaps are still filled following the conventional gap-filling procedure as explained in Chapter 2. Next the source and sink nodes are added. Then the links between the source node and each node in stages 1 are established with zero costs. Similarly, nodes in stage N are connected to sink node with zero costs as well. Dynamic Programming is then employed to find the shortest path (horizon line) in this resultant graph.

3.3 Proposed Nodal Costs

Lie et al. [3] proposed the use of edges where they use binary costs to encode the information in the multi-stage graph about a node being an edge pixel or not. For gap filling, they initialized the dummy nodes with high costs. Although, we reduced the number of horizon candidate edges considerably by

the use of MSEE and trained classifiers yet using only the information about pixels being edge or non-edge is not enough to initialize the nodal costs. It is possible for DP to choose falsely classified horizon edges as part of the solution. To address this, we propose the use of various nodal costs that provide further evidence about the positively classified edges for being true horizon edges. Specifically, we have investigated the following nodal costs.

3.3.1 Gradient Information

As our first proposed nodal cost, we use the information due to gradient magnitudes and difference of gradient magnitudes. Unlike Lie et al.[3] and others who formulated the multi-stage graph based only on edges, we propose dense multi-stage graphs where each pixel would be a node of the graph and connected to it neighbors in the next stage. However, the nodes are initialized according to the gradient information. As gradient magnitudes are used as an intermediate part of edge detection, the DP should find a solution where the sum of the gradient magnitude is maximized. However, maximizing gradient magnitude does not gaurantee a continuous solution. To enforce continuity, we propose a second constraint that the difference between the gradient magnitudes of two adjacent neighbors should be minimized. The two constraints when imposed together result in the detection of the strogest continuous curve.

It is worth noting that the gradient based approach does not involve any kind of training and is presented here to establish the fact that using just the constant low and high costs for edge and non-edge pixels (horizon classified

edges in case of [1] and [2]) are not enough to find the accurate horizons. In gradient based approach; given a query image I, the gradient magnitude for each pixel $I(i,j)$ of the image is computed which can be mathematically expressed by eq. 3.6

$$\nabla(i,j) = \Gamma[I(i,j)] \tag{3.6}$$

Where, Γ is the function which takes a gray scale image as an input and returns the corresponding gradient magnitude image ∇. Next, the difference of the gradient magnitude image is computed. Since, a node i in stage j can be connected to as many nodes as defined by the δ neighborhood; one should generate as many gradient difference images where the difference should be taken with the node in next stage to which the current node is being connected. The equation 3.7 below shows the making of difference of gradient mask for connections at the same level.

$$d\nabla(i,j) = |\nabla(i,j) - \nabla(i,j+1)| \tag{3.7}$$

Since, we want to maximize the gradient magnitude while minimizing the difference of gradient magnitude, we normalize the magnitude and difference images between 0 and 1. The nodal costs $\Psi(i,j)$ of the graph are then set as a weighted combination of these two images depending upon the connection of the current node in the next stage.

$$G_r(i,j) = w_1 * d\nabla(i,j) + (1 - w_1) * (1 - \nabla(i,j)) \tag{3.8}$$

where w_1 is the weight parameter which is set to be $w_1 = 0.5$ in our experiments. Note that since DP solves a minimization problem, we have used the difference $(1 - \nabla(i, j))$ in the equation 3.8. The weighted average G_r was used as the nodal cost:

$$\Psi(i, j) = G_r(i, j) \tag{3.9}$$

The link costs may be initialized using equation 2.3.

3.3.2 Classified Binary Edges

The second nodal cost that we investigate is fairly similar to Lie et al. [3] and others [2] i.e. we use the binary edge information as the nodal cost. The only difference is that, we apply the DP on the graph formulated by reduced number of edges due to MSEE and trained classifier. So, equations 3.3 through 3.5 are used to initialize the graph and to set the node/link costs. We use SIFT-HOG classified edges as we would show in the results that SIFT-HOG concatenation outperforms all other feature choices.

3.3.3 Classified Edge Score

As described earlier, using a fixed low cost for edge pixels provides only partial information with no confidence to compare two positively classified nodes where one might be misclassified. To enforce this knowledge in our dynamic programming formulation, we propose a two fold use of the classifier. First, we use classifier to distinguish between horizon and non-horizon edges as realized by equation 3.2. Secondly, classifier is used to provide a confidence

about an edge pixel being part of the horizon. We normalize the raw scores provided by the classifier between 0 and 1 without using any thresholding. The node costs are then initialized by the actual classification scores instead of initializing all positively classified edges to fixed low cost. The equation 3.4 would be altered to reflect this information. Since, we want to find a shortest path through DP, we assume that the values have been reversed so a smaller value reflects an edge pixel is more probable to be a horizon pixel.

$$\Psi(i,j) = \begin{cases} S(i,j), & \text{if } E_{+}(i,j) = 1. \\ \infty, & \text{otherwise.} \end{cases} \tag{3.10}$$

3.3.4 Classified Edge Score with Gradient Information

In this formulation, we combine the classification information with the gradient information. By fusing equations 3.8 and 3.10, we get a new initialization for the nodal costs;

$$\Psi(i,j) = w_2 * S(i,j) + (1 - w_2) * G_r(i,j), \tag{3.11}$$

where, w_2 is a scalar and we use 0.5 value to weight both the scores equally. Figure 3.7 shows a visual representation of some of the proposed nodal costs.

3.4 Edge-less Method

Our second proposed approach applies DP on a classification map that associates with each pixel a classification score which can be interpreted as the

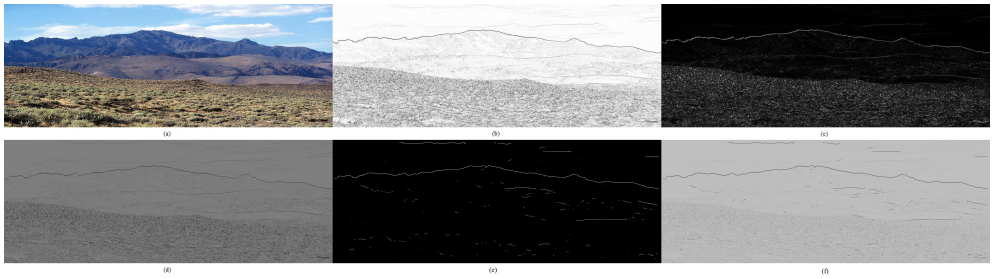

Figure 3.7: Visualization of various types of nodal costs: (a) original image,(b) gradient magnitude image,(c) difference of gradient magnitude image, (d) weighted average G_r, (e) classification score image and (f) classification score image + weighted average G_r.

confidence of the pixel being part of the horizon line. Most importantly, it does not rely on edge detection, therefore, it does not require gap filling or requirement to add dummy nodes as was essential in method proposed by Lie et al. [3] and in our first proposed method as well. Moreover, like our edge-based approach, we do not force the nodes in stages 1 and N to be associated with their vertical positions since the assumption of the horizon line being present in the upper half of the image could be violated (e.g., due to the rover moving on a peak and looking towards the horizon).

Using the trained classifier, a Dense Classification Score Image (DCSI) is generated which is equal to the size of an input image and used as an $M \times N$ multi-stage graph that does not require any node initialization. Once we have introduced the source/destination nodes s/t and decided on the value of δ, any shortest path finding algorithm can be used to find the path that maximizes the sum of classification scores. We will later show that the number of nodes per stage can be significantly reduced by only considering the pixels with the m highest classification scores where m is a parameter; we

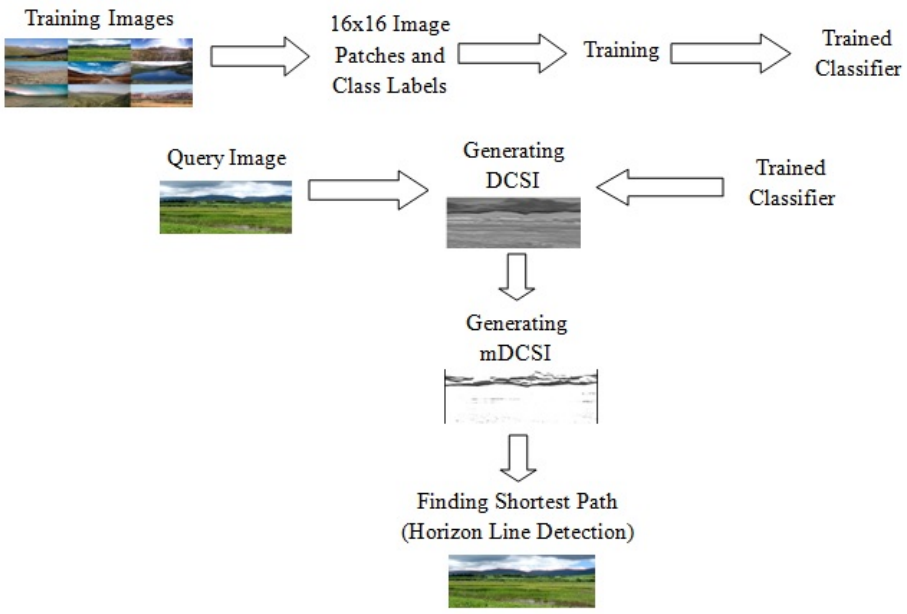

Figure 3.8: Main steps of the training/testing phases of the edge-less approach.

refer to this reduced map as mDCSI map. Using fewer nodes per stage does not affect accuracy while it speeds up computations considerably. Figure 3.8 illustrates the main steps of our edge-less skyline detection approach.

3.4.1 Pixel Classification

For classification, we have experimented with two classifiers: SVM [26] and CNN [27]. Each classifier is trained using horizon and non-horizon image patches from a set of training images where the horizon line has been extracted manually (ground truth) as detailed in section 3.1. Specifically, for each training image, we select N points uniformly from the ground truth; an equal number of points is randomly selected from non-horizon locations as shown in figure 3.3. We take a 16×16 normalized image patch around

each sampled point and the resulted 256-D vector is used for training the classifiers. It is worth mentioning that we are using pixel intensities as features here as compared to texture features that have been used for first method. The decision is partially biased due to the fact that we are training a CNN classifier [27] which is expected to find features itself instead of hand crafted features. For a fair comparison between SVM and CNN classifiers, normalized pixel intensities are used as features for both. We normalize pixel intensities between -1 and 1.

For the CNN classifier, we use an architecture comprising of 2 Convolution(C)-Sub-sample(S) layers. The first C-S layer is comprised of 4 levels with a convolution(C) mask of 5×5 and a sub-sampling(S) mask of 2×2. The second C-S layer is comprised of 8 levels with a C mask of 3×3 and an S mask of 2×2. For the SVM classifier, we use a linear kernel as it was found to be equally good as the Radial Basis Function (RBF) and polynomial kernels. We have only used 9 images for training the classifiers with 343 positive (horizon) and 343 negative (non-horizon) examples extracted from each image. Figure 3.3 shows an example of horizon (red) and non-horizon (blue) training samples. It is important to note that for both approaches, same number of training images and same key points are used, the difference being the feature and classifier choices.

3.5 Detection Steps

This section describes the steps involved for skyline detection through our edge-less method. Figure 3.8 presents a visual overview of these steps.

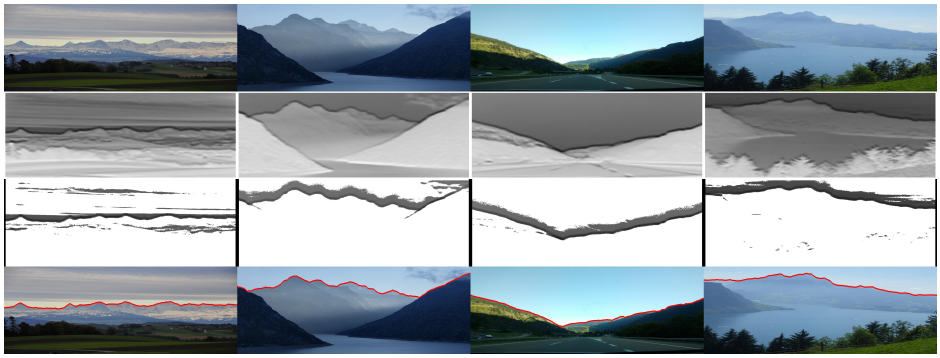

Figure 3.9: Edge-less skyline detection: (row1) sample test images, (row2) respective DCSIs , (row3) mDCSIs, and (row4) detected horizon lines (highlighted in red).

3.5.1 Dense Classifier Score Image (DCSI)

Once the classifiers have been trained, the DCSI can be generated for a given test image. For each pixel location in the test image, a 16×16 patch of pixel intensities around the pixel is extracted. The normalized intensities are then used to form a 256-D vector, which is fed to the classifier. The classification score is then associated with that pixel location. Classification scores are normalized in the interval [0, 1]; the resultant scores form the DCSI which is denoted as $S(i, j)$. In essence, $S(i, j)$ can be interpreted as a probability map which reflects the likelihood of a pixel belonging to the horizon line. Since, no edge are involved; $S(i, j)$ can directly be used to initialize nodal costs as shown in equation 3.12. Figure 3.9 shows the DCSIs for some sample images.

$$\Psi(i, j) = S(i, j) \tag{3.12}$$

3.5.2 Reduced Dense Classifier Score Image (mDCSI)

Although the full DCSI can be used for horizon line detection, we have found that keeping only the m highest classification scores in each column does not compromise the accuracy while reducing computations. This is because the highest classification scores are typically concentrated within a small band around the horizon line. We refer to the reduced DCSI as $mDCSI$. The multi-stage graph corresponding to the mDCSI contains fewer vertices; as a result, fewer paths need to be considered when searching for the shortest path which results in considerable speedups. In our experiments, we have found that by keeping the highest $50-100$ classification scores yields accurate horizon line detections. Mathematically the mDCSI is computed as follows:

$$S_+(i,j) = \begin{cases} S(i,j), & \text{if among m highest scores in col. j} \\ c, & \text{otherwise} \end{cases} \tag{3.13}$$

where, S_+ and S correspond to mDCSI and DCSI respectively. If the i-th pixel (node) in column (stage) j is among the m highest classification scores for column j, the classification score from $S(i,j)$ is used; otherwise, the score is set to a very low score c. Figure 3.9 shows examples of the respective mDCSIs.

3.5.3 Nodal and Link Costs

In earlier approaches [2, 3], the edge map (or classified edge map) was used to form the multi-stage graph which requires gap filling to be an essential

step in extracting the horizon line. Since, this approach does not rely on edge maps, so gap filling is not required. In this approach, the mDCSI is used to create an $M \times N$ graph $G(V, E, \Psi, \Phi)$ with node costs initialized to mD(x,y). So, equation 2.2 can directly be transformed into 3.12 or 3.14 depending upon whether DCSI or mDCSI is used for initialization.

$$\Psi(i, j) = S_{+}(i, j) \tag{3.14}$$

Since the resulted graph is a dense graph, each node i in stage(column) j is connected to three nodes i, $i-1$ and $i+1$ in stage (column) $j+1$ (i.e., $\delta = 1$). These connections are considered as edges with zero costs similar to first approach. However, this is in contrast to conventional approaches ([3]) where the absolute difference between the positions of nodes in two stages is used as an edge cost.

$$\Phi(i, k, j) = \begin{cases} |i-k|, & \text{if } |i-k| \leq \delta \\ \infty, & \text{otherwise.} \end{cases} \tag{3.15}$$

Since, the horizon line might not always appear in the upper half of the image, we do not set the nodes in stages 1 and N proportional to their vertical position. As per [3], two dummy nodes, s and t, are introduced to the left of stage 1 and to the right of stage N respectively. The edge weights from s to every node in stage 1 and from every node in stage N to node t are set to be zero – same as in our edge-based approach. The shortest path in the resultant graph then can be found using DP.

Chapter 4

Experimental Analysis and Fusion

In this chapter, we list the experimental details of our work and highlight how the two proposed methods can be combined into a fusion approach which is more effective. In section 4.1, we outline the details of the data sets being used and the quantitative measure that we have adopted for our experiments. Sections 4.2 and 4.3 provide the results for edge-based and edge-less methods respectively. We identify the failure examples for both Lie et al. and our methods in section 4.4. The strengths and weaknesses of each of our proposed method are discussed in section 4.5. In section 4.5, we further describe the details of proposed fusion method and also compare this fusion strategy with the two proposed methods described in chapter 3.

4.1 Data Sets and Quantitative Evaluation

To evaluate the performance of our proposed approaches, we have experimented with three different data sets: the Basalt Hills data set, Web data set and Switzerland (CH1) data set [17]. The Basalt Hills data set is a subset of a bigger data set which was generated by placing cameras on an autonomous robot navigating through Basalt Hills [23]. We have chosen 45 images from this data set with considerable viewpoint and scene changes. The Web data set is a collection of 80 mountainous images that have been randomly collected from the web. This data set includes various viewpoints, geographical and seasonal variations. The Switzerland – CH1 data set is comprised of 203 images from the Alps, made publicly available by the authors [17] along with the ground truth annotations. We use CH1 as a test set for our proposed fusion approach.

Our training set for both approaches consists of only 9 images from Basalt Hills data set. The resolution of all images in our data sets is 519×1388. In both proposed approaches, same number of positive and negative key points are chosen from each training image. Specifically, we select 343 positive and 343 negative key points from each image in the training set. The positive key points are chosen uniformly from the ground truth skylines whereas the negative key points are chosen randomly from the non-horizon edge locations belonging to both sky (edges due to clouds) or non-sky regions.

To quantitatively evaluate the performance of the proposed approaches, we have manually extracted the horizon line (ground truth) in all the images for both Basalt and Web data sets while ground truths for CH1 data

set have been made available by the authors [17]. To evaluate the proposed approaches, the detected and true horizon lines are compared by calculating a pixel-wise absolute distance between them. For each column, the absolute distance between the detected and ground truth pixels is computed and summed over the entire number of columns in the image. The resultant distance is normalized by the number of columns in the image, yielding the average absolute error of the detected horizon line from ground truth. Since nodes in a particular stage are not allowed to be connected to nodes in the same stage, so the true and detected horizon lines are bound to have the same number of columns/stages in the image/graph. Hence, there is a one-to-one correspondence between the pixel locations in the true and detected horizon pixel locations. Mathematically, the pixel-wise absolute distance S between true and detected horizon lines can be described as,

$$S = \frac{1}{N} \sum_{j=1}^{N} |P_{d(j)} - P_{g(j)}| \tag{4.1}$$

where $P_{d(j)}$ and $P_{g(j)}$ are the positions (rows) of the detected and true horizon pixels in column j respectively and N is the number of columns in the given image. We have evaluated our two proposed methods on the Basalt Hills and Web data sets. Both methods are also compared against the traditional edge based approach of Lie et al. [3]

4.2 Results for Edge-based Method

4.2.1 Effect of MSEE

As a first step to our edge-based proposed approach, we compute MSEE images for all the images in our data sets. We compare the number of edges survived after MSEE with the number of edges found by Canny edge detector [30] of Matlab. We see a considerable reduction in the number of candidate horizon edges which are further reduced by the classifier. Table 4.1 shows the average percentage reduction in number of edges for both Basalt Hills and Web data sets.

Table 4.1: Importance of MSEE

Data Set	**Average % Reduction**
Basalt Hills	66.37
Web	43.45

4.2.2 Effect of Texture Descriptors

For our first approach, we have investigated various texture descriptors and their combinations as the feature choices to train SVM classifiers. To compare these feature descriptors for skyline detection problem, we have performed a 5-fold cross validation on the Basalt Hills data set. For each fold, we divided the data set into non-overlapping train (9 images) and test sets (36 images). Since, we have the ground truth horizons at our disposal, therefore, we know which edges belong to true horizon and which do not. Table 4.2 shows the percentage False Positive (FP) and False Negative (FN) errors averaged over

the five folds of training and the respective standard deviations. Figure 4.1 shows a graphical view of the same information. Since, false negative error is of more importance, we choose the classifier based on SIFT-HOG combination for further evaluation, highlighted in table 4.2 .

Table 4.2: % FP and %FN errors for various features

Feature	**%FN**		**%FP**	
	μ	σ	μ	σ
SIFT	1.0224	0.7890	17.8090	5.6089
LBP	5.0332	8.3747	10.6366	8.0770
HOG	2.3285	2.3498	11.2331	5.6616
SIFT+LBP	0.6915	1.1827	10.6065	5.8949
SIFT+HOG	**0.6624**	**0.7436**	**11.0801**	**5.1797**
LBP+HOG	3.3647	3.6415	10.0737	6.394
SIFT+LBP+HOG	7.1887	8.1177	4.8302	4.0438

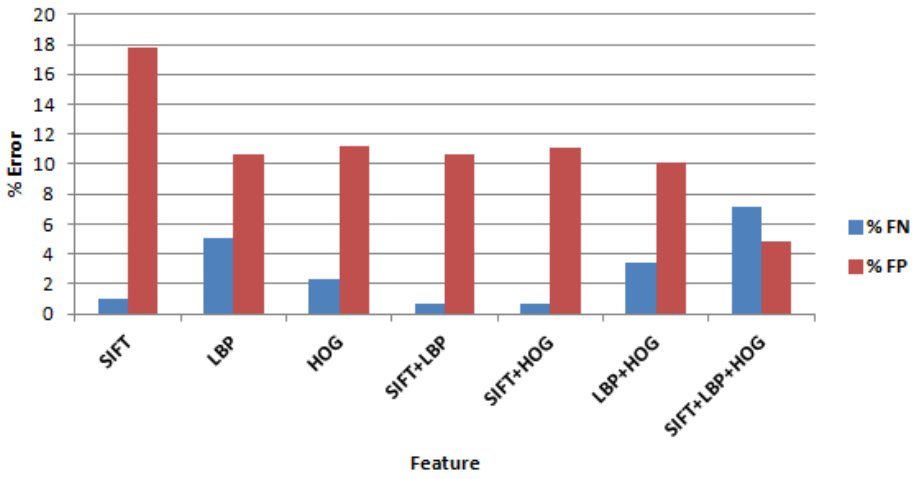

Figure 4.1: Mean of % false positive and false negative errors for various features

Table 4.3: Edge-based Approach: Average Absolute Errors

Nodal Costs	**Basalt Hills**		**Web**	
	μ	σ	μ	σ
Lie et al. [3]	5.5548	9.4599	9.1500	17.9195
G_r	3.9908	6.3530	11.8641	26.8084
SIFT+HOG Edges	0.5783	1.0227	**0.8698**	**1.0366**
SIFT+HOG Scores	**0.4124**	**0.8120**	0.9704	1.5698
SIFT+HOG Scores + G_r	0.4358	0.8124	1.3016	3.9814

4.2.3 Best Nodal Cost

We compare our proposed formulations (chapter 3) with the state of the art horizon detection method based on edges and DP i.e. the approach by Lie et al. To compare the detected horizon lines found by each method with the ground truth horizons, we compute an average pixelwise absolute error using equation 4.1. Since, in all of our formulations, we do not allow nodes to be connected within the same stage, there exists a one-to-one mapping between the pixels of the detected (d) horizon and the ground (g) truth horizon. For each of our formulations and Lie at al. [3]; we compute the average and standard deviation over all images in the data sets listed in table 4.3. Clearly, SIFT+HOG Scores outperforms all others strengthening our understanding that using only edge information is not enough for the nodal costs.

4.3 Results for Edge-less Method

4.3.1 Quantitative Evaluation

For the second proposed method, the results have been computed for both Basalt and Web data sets mentioned earlier and same absolute average error

Table 4.4: Edge-less Approach: Average Absolute Errors

Approach	**Basalt Hills**		**Web**	
	μ	σ	μ	σ
Lie et al. [3]	5.55	9.46	9.15	17.92
G_r	3.99	6.35	11.86	26.81
SVM-mDCSI	1.01	0.29	1.28	1.20
CNN-mDCSI	0.75	0.23	1.41	1.49

is computed as listed in equation 4.1. Table 6.1 shows the average absolute error for all the images in each data set, both for the SVM and CNN classifiers. For comparison purposes, we also provide results based on the method of Lie et al. [3] and the gradient information. It is interesting to note that although our methods were trained using a very small number of images from the same data set, yet they generalize well to images from other data sets, such as the Web data set which is very different from the training data set.

4.3.2 Comparing Classifiers

Comparing the two classifiers used in our experiments for edge-less approach, the CNN classifier outperforms the SVM classifier for the Basalt Hills data set while SVM outperforms CNN on the other data set. This indicates that the features found by the CNN classifier might not generalize well to different data sets. Figure 4.2 shows some representative DCSI results using the SVM and CNN classifiers. It is worth noting that the CNN classifier provides a crispier DCSI, having a narrower band around the true horizon line as compared to the DCSI produced by SVM. It might be possible to further improve our best results by combining the SVM and CNN classifiers but we

have not experimented with this idea.

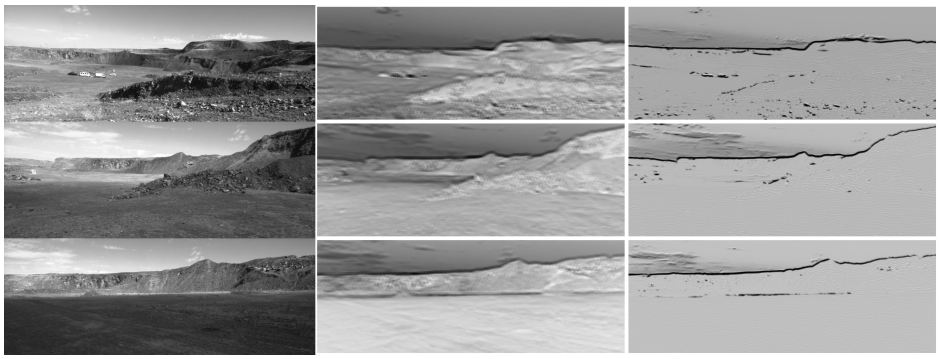

Figure 4.2: Comparison of the classifiers: (column1) Test images, (column2) corresponding SVM-DCSIs and (column3) corresponding CNN-DCSIs.

4.4 Discussion

4.4.1 Failure of Lie et al. [3]

Our experimental results illustrate that the proposed approaches outperform the traditional approach of [3] based on edge maps. In particular, both average error and standard deviation of the traditional approach are much higher than the proposed approaches based on SVM or on both SVM and CNN classifiers in edge-less approach. To better illustrate the performance of the traditional approach, we have identified specific examples where it fails to detect the true horizon line or it misses parts of the horizon. The main reason for this is due to the presence of big gaps in the edge map. This might happen due to different reasons, for example, horizon edges might not be strong enough or stronger edges might exist close to the horizon line due to various environmental effects such as clouds. Although Lie et al.

have proposed a gap filling approach by introducing dummy nodes with high costs. This gap filling approach also have limitations. For instance, when gaps are long and edges from the clouds are close to the horizon line, the gap filling approach fails to fill these gaps. In such cases, it is likely that the DP approach might find a low cost path by taking an alternative path. Figure 4.3 (row 1) shows two examples where the method of Lie et al. has failed to find a good solution due to edge gaps and the presence of clouds, however, our proposed method (edge-less) was able to find the true horizon line with high accuracy in both cases (row 2). Figure 4.4 shows zoomed sub-images of the left column images of 4.3 (i.e., Lie et al.) for better visualization.

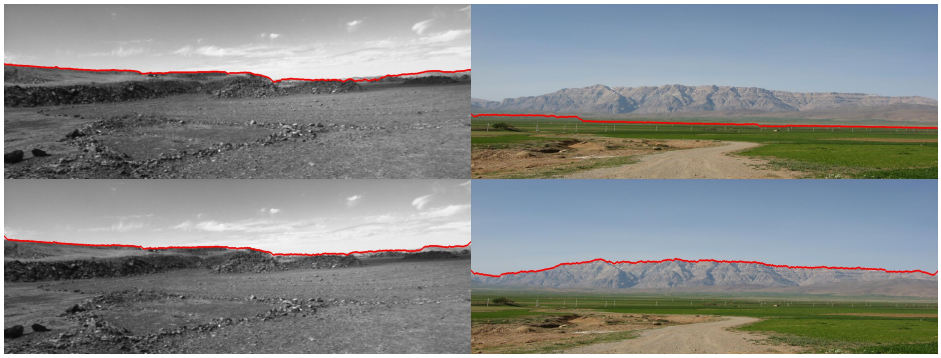

Figure 4.3: Examples illustrating: (row1) missing the horizon line or parts of it due to edge gaps (Lie et al.), and (row2) detecting the true horizon line using our edge-less method (SVM).

Another reason affecting the performance of Lie et al. is the underlying assumption of the horizon boundary is close to the top of the image. When clouds are present in an image, a portion of the true horizon may be missed if the true horizon line is below the clouds due to the bias introduced towards solutions closer to the top of the image (equation 2.4). Figure 4.5 shows some

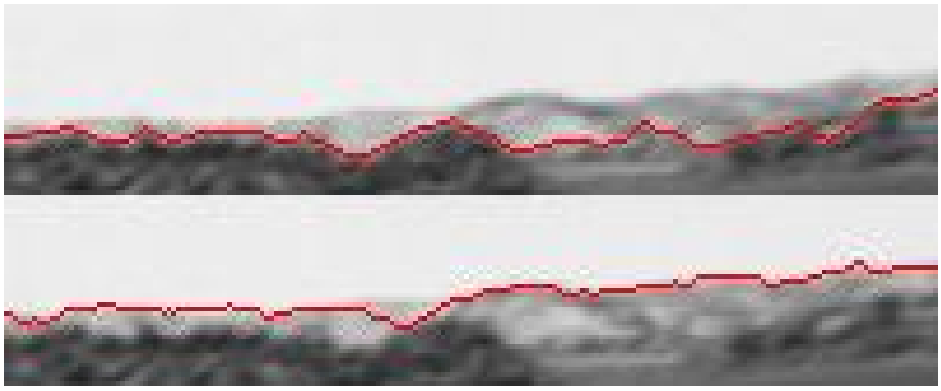

Figure 4.4: Zoomed sub-images of the left column images of Figure 4.3

examples where the approach of Lie et al. has found solutions consisting of both horizon line segments as well as cloud edge segments. Our approach, on the other hand, was able to find the correct solution for these cases as it does not make such assumptions.

Figure 4.5: Examples illustrating: (row1) missing parts of the horizon line due to the assumption that the horizon line is close to the top of the image (Lie et al.), and (row 2) detecting the true horizon line using our edge-less method (SVM).

4.4.2 Failure of Proposed Approaches

In an Attempt to better understand why the proposed approach (edge-less) sometimes fails to give optimal solutions is primarily because of two reasons. First, restricting intra-stage node connectivity and allowing succeeding stage node connectivity. In the multi-stage graph formulation of Lie et al., a node at stage j is only allowed to be connected to nodes at stage j+1 which is problematic when the horizon line has high slope (i.e., steep peaks). Figure 4.6 (row 1) shows an example in which a mountain with a steep peak is shown and Lie et al., approach fails to detect the horizon correctly. This issue can be easily rectified by allowing to make intra as well as succeeding stage node connectivity. Figure 4.6 (row 2) shows the solution obtained by allowing connections within the same stage. However, this improvement is at the expense of increased computational cost as the number of paths that needs to be explored are more.

The second main reason affecting the performance of the proposed approach is due to the use of a very small set of training images (i.e., only 9 images from the same data set). Figure 4.7 shows some examples where the proposed method has failed to find good solutions. This issue can be addressed by increasing the size of the training set and making it more versatile. Careful analysis of our results on the Web data set shows that the proposed approach failed to find a good solution in 9 out of the 80 images due to using a small training set. For our case, we have defined a good solution which has the average absolute error below 1.5 pixel. Removing these images from the data set improves the average error of our approach using the SVM classifier

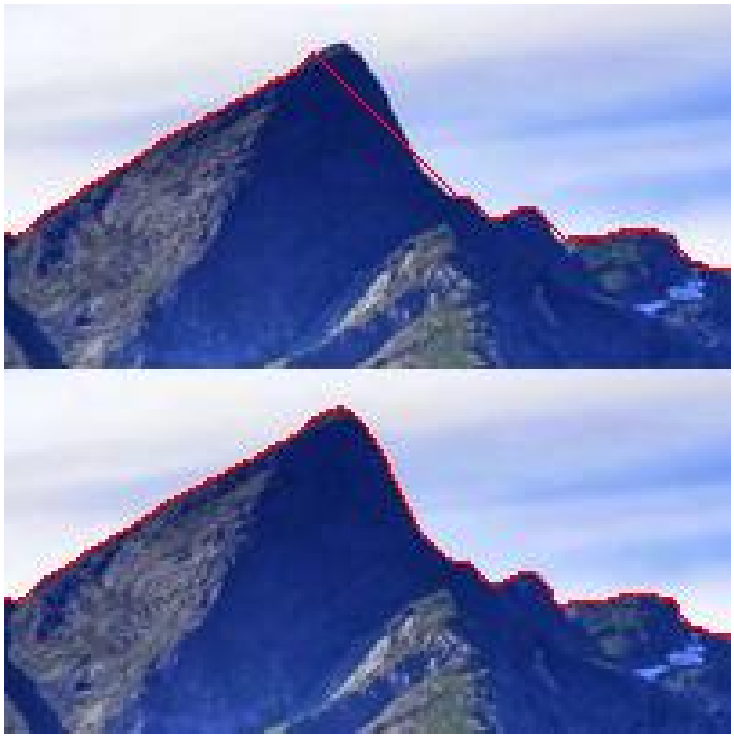

Figure 4.6: (row1) effect of not allowing node connections within the same stage; (row2) solution obtained by allowing node connections within the same stage.

from 1.2854 to 0.9227 while the variance is reduced from 1.1988 to 0.3637 i.e. a sub-pixel accuracy is achieved for 90% of the images.

Figure 4.8 shows some representative results of our edge-less approach using images from both data sets.

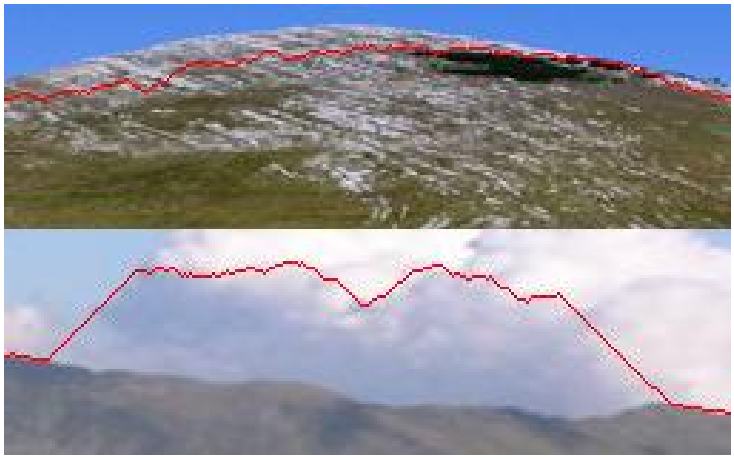

Figure 4.7: Examples illustrating the inability of the proposed method to find a good solution due to the lack of sufficient training data.

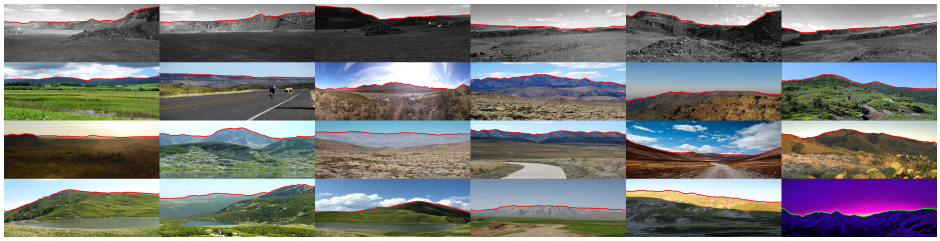

Figure 4.8: Sample results illustrating our edge-less skyline detection approach: Basalt Hills data set (row1) and Web data set (row 2 through 4). Detected horizon lines are highlighted in red/green.

4.5 Fusion of Edge-less and Edge-based

4.5.1 Edge-less versus Edge-based

For comparison purposes, Table 6.2 compares the proposed edge-less approach with the edge-based approaches using edge classification. As it is evident from Tables 6.1 and 6.2; using machine learning for horizon line detection is very promising. Both machine learning approaches outperform the

Table 4.5: Average Absolute Errors Using Edge-based (with classification) and Edge-less Approaches

Approach	**Basalt Hills**		**Web**	
	μ	σ	μ	σ
SIFT+HOG Edges	0.57	1.02	0.87	1.03
SIFT+HOG Scores	0.41	0.81	0.97	1.57
SIFT+HOG Scores + G_r	0.43	0.81	1.30	3.98
SVM-mDCSI	1.01	0.29	1.28	1.20
CNN-mDCSI	0.75	0.23	1.41	1.49

classical edge-based approach [3] by a high margin. A quick look of Table 6.2 reveals that our edge-based approach outperforms the edge-less approach by a small margin. In an effort to better understand and explain these results, we have identified several reasons which are discussed below. Moreover, to leverage the strengths and weaknesses of each approach, we propose a fusion strategy.

4.5.2 Ground Truth Bias and Multiple Horizons

While generating the ground truth, we used the results of edge detection to better localize the horizon line which favors the edge-based methods when computing the error. Moreover, sometimes there are more than one horizon lines in an image (e.g., lower mountains sitting in front of higher and distant mountains); while generating the ground truth in those images, we chose the strongest edge segments. When using edge-based methods for horizon line detection, these segments are typically part of solution; however, this might not be the case for edge-less methods.

Figure 4.9: Edge-less (left column) vs edge-based (right column) horizon detection results: (a) multiple horizons, (b) smoother localization, (c) miss-classifications.

4.5.3 Smoother Localization

Edge-based solutions tend to be smoother while edge-less method are typically bumpy. This is because DP tries to find a path with a low cost without imposing any smoothness constraints on the solution which favors edge-based methods again.

4.5.4 Miss-Classifications

Edge-less methods suffer more from miss-classifications compared to edge-based methods. This is because every pixel is classified in the case of edge-less methods while only a much smaller number of edge pixels are classified in the case of edge-based methods.

Figure 4.9 shows several examples where edge-based horizon detection has outperformed edge-less horizon detection. Figure 4.10 provides more details.

4.5.5 Fusion

We discussed in the previous section several reasons favoring edge-based methods. On the other hand, the main advantage of the edge-less approach is that the DCSI map contains no gaps. To improve horizon line detection, we propose fusing information from edge-based and edge-less methods.

The fusion of gradient information with pixel classification scores is a natural extension of equation 3.11 assuming that both G_r and S are dense maps. Fusing edge information with pixel classifications can be performed in two steps. First, the DCSI map (i.e., $S(i, j)$) is generated for the query image which provides the horizon-ness for each pixel. Second, edge detection

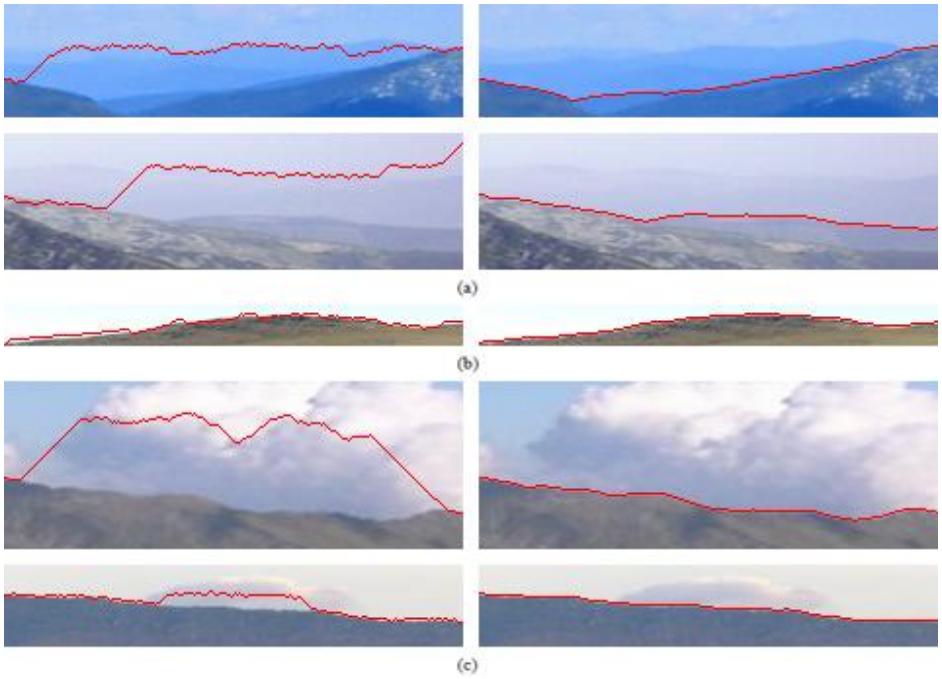

Figure 4.10: Detailed segments from Figure 4.9 : (a) multiple horizons, (b) smoother localization, (c) miss-classifications.

Table 4.6: Average Absolute Errors Using Fusion of edge-based (No Classification) and edge-less Information

Approach	**Basalt Hills**		**Web**	
	μ	σ	μ	σ
SIFT+HOG Edges	0.57	1.02	0.87	1.03
SVM-mDCSI	1.01	0.29	1.28	1.20
SVM-DCSI+G_r	0.60	0.29	4.86	15.98
SVM-DCSI+MSEE Edges	0.73	0.32	0.85	0.89
SVM-DCSI+Canny Edges	0.77	0.35	0.78	0.76

is performed on the query image; then, the horizon-ness of each pixel is boosted if that pixel happens to be an edge pixel. We have considered both Canny and MSEE edges in our experiments. Depending on whether Canny edges (i.e., $E(i,j)$) or MSEE edges (i.e., $E_m(i,j)$) are used, equation 3.12 needs to be modified as follows:

$$\Psi(i,j) = \begin{cases} S(i,j) + b, & \text{if } E(i,j) = 1 \\ S(i,j), & \text{otherwise;} \end{cases} \tag{4.2}$$

$$\Psi(i,j) = \begin{cases} S(i,j) + b, & \text{if } E_m(i,j) = 1 \\ S(i,j), & \text{otherwise;} \end{cases} \tag{4.3}$$

where, b is a constant added to strengthen the horizon-ness of a pixel. Once the nodal costs have been assigned, the rest of the steps (i.e., link costs and DP) are the same as described earlier. Figure 4.11 shows various steps of the proposed fusion strategy. The DCSI 4.11-(b) is combined with MSEE edges 4.11-(c) to obtain the fused DCSI 4.11-(d) which is then used for finding the shortest path (horizon) by applying DP.

Figure 4.11: Fusion based horizon detection: (a) query image, (b) DCSI, (c) MSEE Edges, (d) fused DCSI, (e) detected horizon (red boundary).

Table 6.3 provides a quantitative comparison of our fusion formulations. For completeness, we have also included the best results of the edge-based and edge-less approaches from Table 6.2. As Table 6.3 shows, using gradient information tends to harm the overall accuracy which is in agreement with our earlier results using gradient information (Table 6.1). This is because there are edges, due to clouds, with strong magnitudes near the horizon which become part of the DP solution. Ignoring the strength of edges (i.e., gradient magnitude) by simply boosting the horizon-ness of edge pixels is more effective in excluding cloud edges from becoming part of the detected horizon line. It is interesting to note that using Canny edges for fusion is slightly better on the Web data set than using MSEE edges; this is because some horizon edges might not survive during the extraction of MSEE edges. Overall, the proposed fusion approach (**SVM-DCSI+Canny Edges**) has

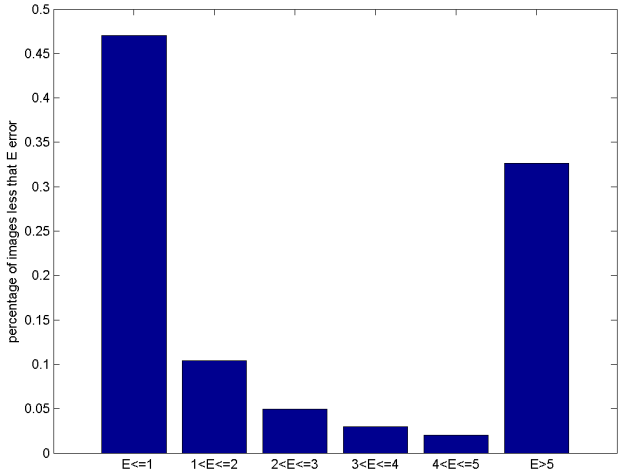

Figure 4.12: Percentage distribution of average absolute error across images in Switzerland data set [17]: 45% horizons detected with sub-pixel error while 67% solution with an average absolute error of less than 5-pixels.

outperformed all edge-based and edge-less formulations.

4.5.6 Further Evaluation of Fusion Approach

To further test the fusion approach, we have considered the Switzerland data set [17]. This data set is comprised of more than 200 mountainous images with considerable viewpoint, terrain and weather variations. The ground truth has been made available by the authors of [17]. Our fusion strategy has achieved a sub-pixel average error in 90 out of the 203 images and less than 5 pixels error in 67% of the images. Figure 4.12 shows the distribution of absolute average error for the Switzerland data set. It should be emphasized that these images are very challenging, not part of the data set used to train the classifier, and captured under different seasonal conditions

Figure 4.13: Examples of faulty detection with average error more than five pixels: sample images (column 1 & 3), solution found by fusion approach (column 2 & 4).

and geographical locations. It is worth mentioning that Baatz et al. [17] have reported that human interaction was necessary for almost half of these images in order to extract the horizon line in their localization experiments. Also, they did not report the average errors for the detection; so it is not clear how good the detection was for half of the images where horizon was detected without any human involvement. Figure 4.14 shows some examples from the Switzerland data set, along with the ground truth and detected horizon lines using our fusion method (SVM-mDCSI+Canny Edges) while Figure 4.13 shows some examples of faulty detections with average absolute error beyond five pixels.

It should be noted that in most of these cases, DP found a solution which corresponds to discontinuity caused by another mountain sitting in front

of a distant (less discontinuous) horizon or snowy mountains having more confidence than the actual horizon. The fusion approach did not get any benefit from edges because both faulty and actual horizon has edge support where as the solution has more confidence due to classifier compared to the actual horizon. It should be noted that our training set of nine images does not contain a single snowy image. We expect the accuracy to increase a lot if snowy examples are added to the training set.

4.5.7 Verification

The distribution of the nodal costs along the DP solution (i.e. detected horizon) provides a metric about the good-ness of the solution. We exploit this information to verify if a found solution is actually a good/acceptable solution (error $<=$ 5 pixels) or a faulty one (error $>$ 5 pixels). This verification can be of great interest; for example in visual geo-localization or planetary rover localization; knowing that the found horizon solution is a faulty one would save the computations in the subsequent steps of localization pipeline i.e. DEM rendering and horizon matching etc. Specifically, we compute the mean and standard deviation of the nodal costs for the pixels belonging to the detected solution by DP. The mean and/or standard deviation measure the divergence of the nodal costs for the nodes belonging to horizon solution. The intuition being that: for a faulty solution the nodal costs along the path would diverge and hence result in comparatively big values for mean and standard deviation. A simple 1d/2d Gaussian classifier can then be trained for the verification of found solutions. The computation for mean (μ_d) and

Table 4.7: Percentage verification accuracies for different 1d/2d Gaussian classifiers for the good-ness of [17] data set.

Gaussian Classifier	**TP**	**TN**	**FP**	**FN**
1D : μ vector	92.42	50.75	49.25	7.57
1D : σ vector	93.18	82.09	17.91	6.81
2D : μ & σ vectors	89.39	82.09	17.91	10.60

standard deviation (σ_d) are shown in the equations below,

$$\mu_d = \frac{1}{N}\sum_{j=1}^{N}\Psi(P_{d(j)}, j) \tag{4.4}$$

$$\sigma_d = \sqrt{\Big(\frac{1}{N}\sum_{j=1}^{N}\big[\Psi(P_{d(j)}, j) - \mu_d\big]^2\Big)} \tag{4.5}$$

where $P_{d(j)}$ is the node (row) index for jth stage (column) in the detected path, N is the number of stages in the graph for DP and cost Ψ follows from equation 4.2.

For verification of horizon solutions found for [17] data set, we first computed the μ_d and σ_d for all the images in the Web data set and formulated 1d Gaussian classifiers based on means, standard deviations vectors alone and a 2d Gaussian classifier based on both mean and standard deviation vectors. These classifiers are then used to verify if a found solution is good or faulty based on computed mean and deviation for each solution from Switzerland data set [17]. Table 4.7 shows the percentage false positive, false negative, true positive and true negative rates for each of these 1d/2d classifiers used to verify solutions from [17] data set. To the best of our knowledge this is the first attempt towards measuring the good-ness of a found horizon solution.

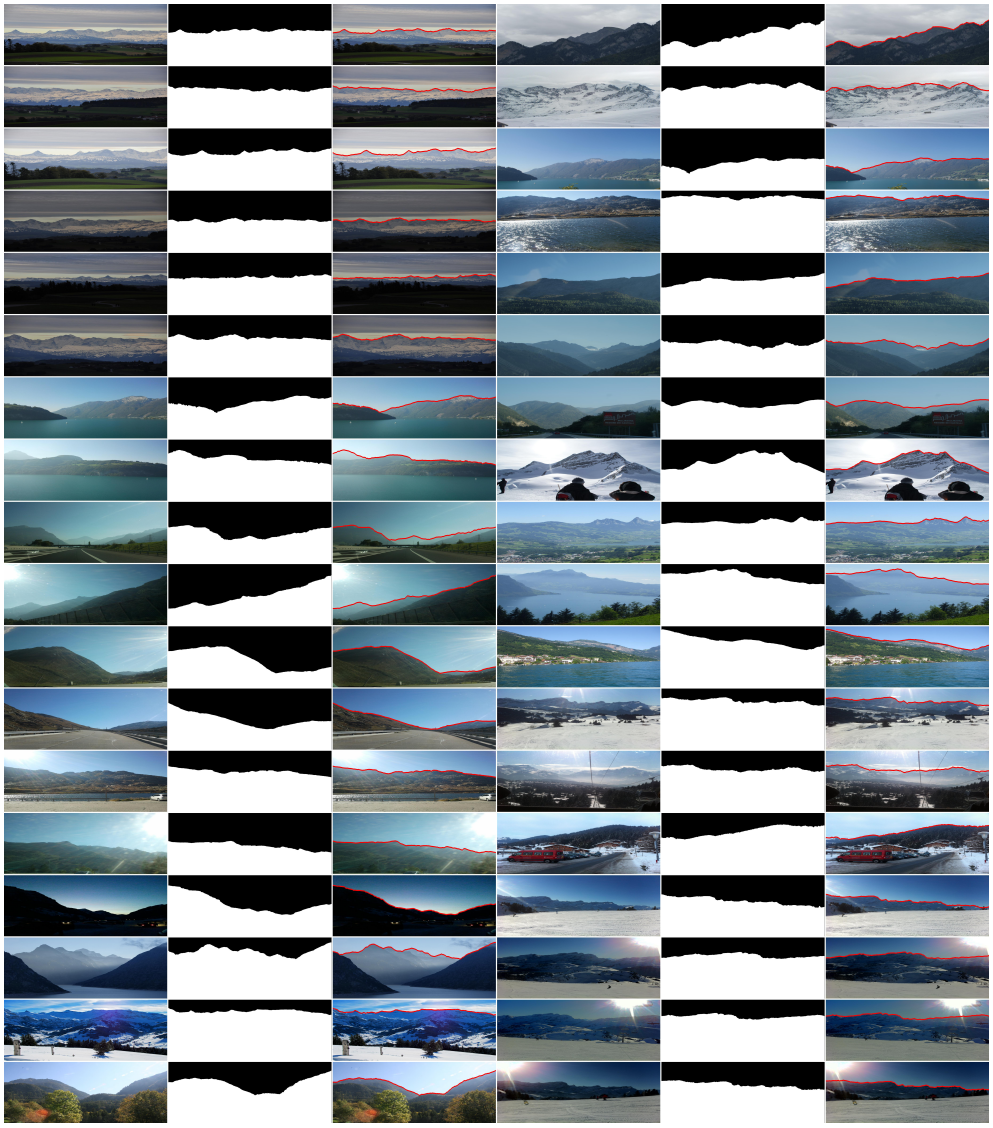

Figure 4.14: Results of the proposed fusion on sample test images from Switzerland data set [17]: query images (columns 1 & 4), ground truth segmentations (columns 2 & 5) and found horizon lines (columns 3 & 6, highlighted in red).

Chapter 5

Extensions of the Proposed Approach

Our proposed skyline detection methods can be adopted for relevant important problems in autonomous planetary and aerial robotic platforms. For example, confirming the presence of skyline in a scene can be very useful for a navigating planetary rover as available limited computational power can be used to accomplish different tasks instead. Similarly, partial horizon can still be used for flying and terrestrial robots' navigation. These capabilities of our proposed skyline detection methods are demonstrated in this chapter. In section 5.1, we demonstrate how the absence of skyline can be detected as a by product of DCSI. Section 5.2 describes the additional components required to make our skyline detection method capable of detecting the partial horizons.

5.1 Non-Horizon Line Detection

The solution verification introduced in previous chapter (section 4.5.7) can further be extended for verifying the presence/absence of horizon. In various applications, including rover navigation, it is important to detect the horizon line with high confidence. In the DP formulation, however, a shortest path solution can be found irrespective of whether the horizon is actually present in the query image or not. To the best of our knowledge, this problem has not been addressed before in the literature either. To determine the presence/absence of the horizon line, the mDCSI map can be used again under the same framework identified in section 4.5.7. Specifically, a narrow band of pixels with high horizon-ness (i.e., classification scores) is typically found around the true horizon line; the absence of such a narrow band will indicate that the horizon line is not visible as shown in Figure 5.1. In this case, the shortest path found will have extortionate cost than a typical solution where the cost is defined as the sum of classification scores along the path found. A simple Gaussian classifier can then be used to verify if the proposed solution reflects absence of horizon in the query image same as used to verify the good-ness of detected solution in previous chapter.

5.1.1 Results

Using a collection of 40 mountainous images from web where horizon line was not visible, we were able to confirm that Gaussian classifier based on Web data set correctly classified these solutions to be faulty. Some of these examples along with their DCSIs, mDCSIs and found solutions are shown

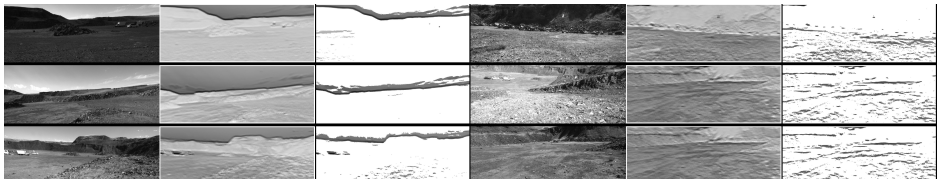

Figure 5.1: Absence of horizon affecting the mDCSIs, causing them to have high scores for shortest paths: Column 1 and 4 showing images with and without horizons respectively. Respective SVM-DCSIs (columns 2 & 5) and mDCSIs (column3 & 6) show no continuity for non-horizon images.

in figure 5.2. It is worth noting that how DP is trying to find a continuous path e.g. along the mountains, pathways and roads. The DCSIs generated are based on the original proposed edge-less approach and not that of fusion and hence the respective nodal costs used for the Gaussian classifier.

5.2 Partial Horizon Line Detection

Previous studies have not addressed the issue of partial horizon line detection i.e. when segment of the horizon is missing and horizon not necessarily extends from left most column to right most. When flying and ground robots move at various angles or steep terrains, it is often the case that only a partial horizon line is visible. From a theoretical point of view, the presence of a partial horizon line provides sufficient information in a number of tasks, for example, in robot localization and visual geo-localization. The problem of partial horizon detection boils down to the problem of determining where to place the start and sink nodes in the DP formulation. In the general case, it is assumed that the horizon line extends from the left most column to the right most column of the image which might not always be true as shown in

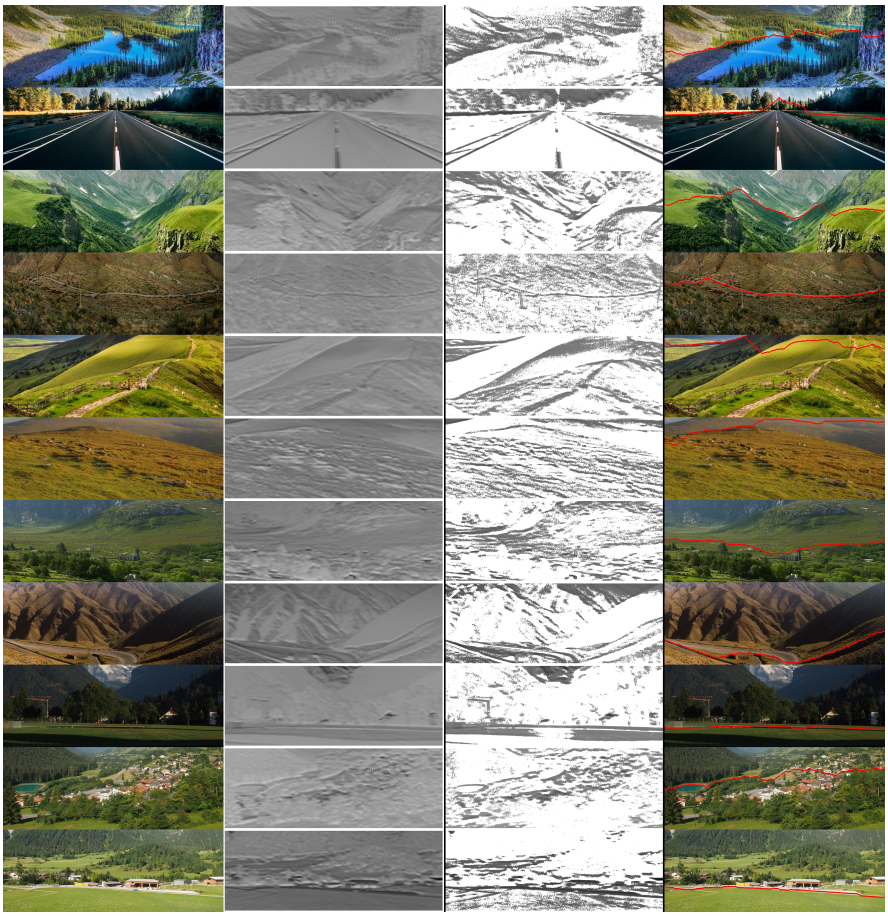

Figure 5.2: Examples of absence of horizon line detection: sample images (column 1), respective DCSIs (column 2) and mDCSI (column 3) and faulty solutions found by DP (column 4, highlighted in red) which would be identified by the Gaussian classifier as faulty detections.

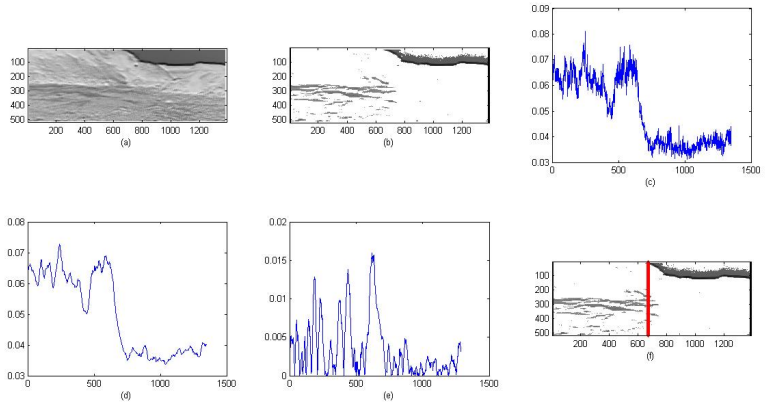

Figure 5.3: Steps towards determining the start/end point for partial horizons: (a) DCSI for an image with partial horizon, (b) mDCSI, (c) average classification scores for each column, (d) smoothed averages, (e) peak corresponding interest point and (f) found end point marked by vertical red bar.

Figure 5.4 (col2). The mDCSI map can be exploited to determine the end points of the horizon boundary. Given a query image, the mDCSI map is computed as described earlier. Then, we compute an average classification score for each column in the mDCSI map and apply smoothing (based on averaging) using a 1x3 window. After smoothing, the start/end points of the horizon line are determined by local maxima detection. Figure 5.3 shows a sample DCSI image containing a partial horizon from the Basalt Hills data set (a), the respective mDCSI (b), the average and smoothed classification scores (c,d), the end point detected through local maxima detection (e), and the found end point marked on the mDCSI (f).

5.2.1 Results

We have experimented with 40 partial horizon images collected from web and were able to successfully find the partial horizon in 35 of them. Some representative results of partial horizon line detection are shown in Figure 5.4 along with the end points found by our method. It should be mentioned that very accurate detection of the start/end points is not necessary and that extracting a portion of the horizon line would be sufficient for various applications. We are again demonstrating these results based on edge-less approach/nodal costs and not the fusion strategy, although that can be equally applicable.

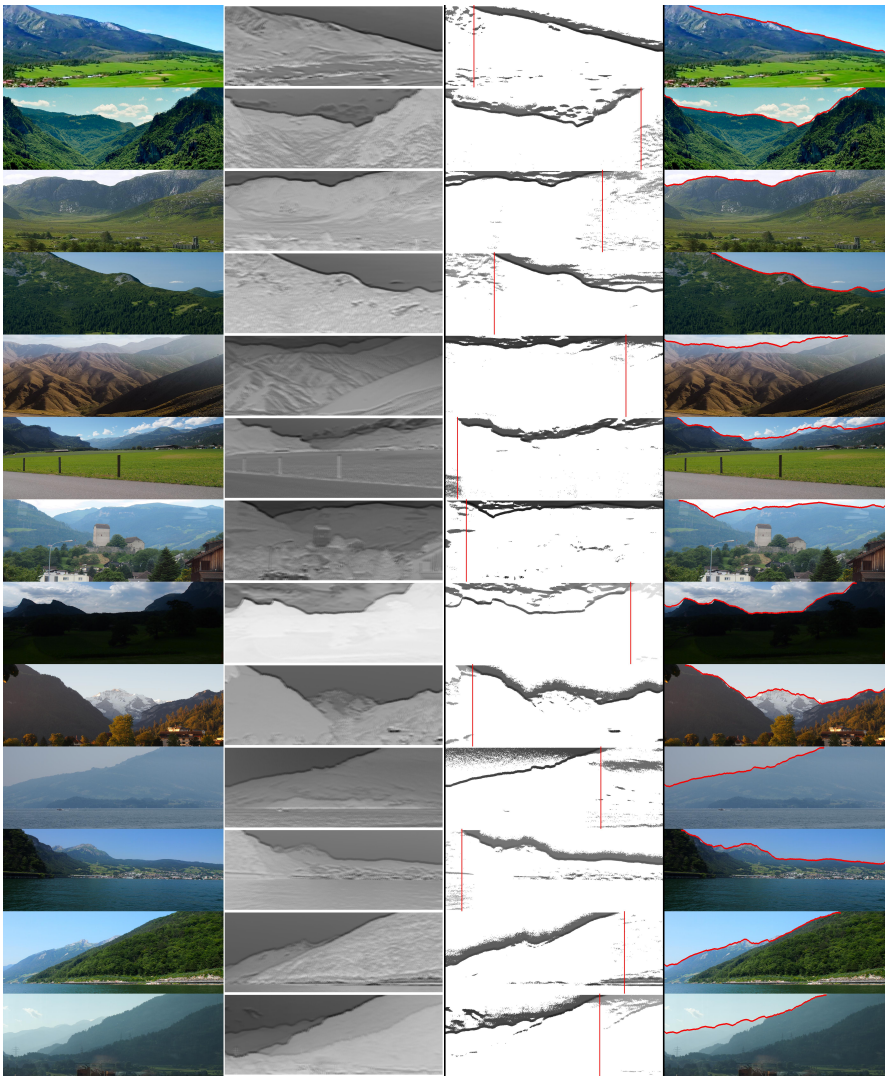

Figure 5.4: Examples of partial horizon line detection: sample images (column 1), respective DCSIs (column 2) and mDCSI (column 3) with determined start/end points (marked by vertical red bars) and partial horizons found by DP (column 4, highlighted in red).

Chapter 6

Skyline Detection using Deep Networks

In the last five years, Deep Learning has emerged as a new dimension in machine learning and have been successfully applied to various problems in computer vision and specifically object detection and recognition. The core to these methods are massive Convolutional Neural Networks, large amounts of training data and computational resources. The underlying training algorithm i.e. back propagation have been around since 1960 and was successfully used by LeCun for digit recognition in 1980s.

Naturally, we have adopted two of the most popular deep neural networks for skyline detection problem and have compared these nets with our method. This chapter provides the details of a comparison of our edge-less skyline detection approach with three other methods – two deep networks and one instance of classical style machine learning method. In section 6.1, we list the technical summary for each of these methods. Section 6.2 describes

the specific details regarding training or otherwise for each of the compared approaches. The results and experimental details are then listed in section 6.3.

6.1 Compared Skyline Detection Methods

6.1.1 Automatic Labeling Environment (ALE)

The skyline approach by Saurer et al. [33] is also based on dynamic programming however the energy function being minimized is more involved. They try to incorporate both data and smoothness constraints in the cost function. They formulate the problem as foreground(non-sky)-background(sky) segmentation problem where a per-column highest foreground candidate is searched subject to minimization of data term and smoothness term in the energy function. The data term in one column evaluates the cost of all pixels below the candidate to be assigned to foreground class and all pixels above above to it are assigned to the background class. The pixel-wise likelihoods are computed through the classifier trained on contextual and super pixel representation. The smoothness term is based on the assumption that all pixels on the contour should have a gradient orthogonal to the skyline. Their pipeline also allows the user to mark foreground/background strokes for challenging images where all the pixels above the marked stroke are assigned to background and those below the stroke to the foreground. Their training was based on 203 images from the CH1 [17] set and testing was done using 948 images from CH2 set [33].

6.1.2 Fully Convolutional Neural Networks (FCNs)

Long et al. [42] built fully convolutional neural network (FCN) that is able to semantically segment image into multiple classes. The network can take input image of arbitrary size and produce semantic labeling of corresponding size i.e. end-to-end training. This model exceeded other state-of-the-art methods for semantic segmentation. The authors adapted several structures of neural network models that were used for classification tasks (VGG net [48], GoogLeNet[49], AlexNet[47]) and fine-tuned them for the segmentation task. The networks were evaluated on PASCAL VOC, NYUDv2 and SIFT Flow datasets, where they achieved state-of-the-art results for multiclass segmentation. Specifically the conventional fully connected layers towards the right end of these networks are replaced with convolutional layers. The core of the FCN is the skip-layer architecture which combines the deep, coarse semantic information with shallow, fine appearance information.

Figure 6.1 shows the VGG network [48] transformed into FCN32s. Each of the convolutional (conv) layer is followed by an element-wise Rectified Linear Unit (ReLU) and a dropout layer (only in conv6 and conv7); color codings are provided to note the specific differences. For all the conv layers the receptive field, zero-padding and stride are of 3, 1 and 1 respectively except where explicitly noted as (F/P/S) to the left of the conv layer e.g. conv1_1 has a padding of 100 and conv6 has a kernel of 7. Each module of the conv layers is followed by a max pooling layer with a filter of size 2 and stride 2. The number of output channels for each of the conv module are noted with the number next to the arrow emerging from preceeding pooling

layer. Conv layers conv6 and conv7 are each followed by a dropout layer with 50% ratio i.e. 50% of the neurons are dropped randomly in the respective layers to ensure generalization [51]. The conv8 layer is the compression layer responsible for compressing the 4096 channels to N channels where N is the number of classes. The de-convolution layer (de-conv) performs up-sampling while crop layer takes two inputs and crop the first according to the dimensions of the second. The softmax loss function is used to guide the stochastic gradient descent which takes the output of the crop layer and semantic label equivalent to the size of images.

Given the input resolution of a conv layer, the resolution of the output can be computed using Eq. 6.1 while Eq. 6.2 can be used for similar calculation for the de-conv layer; I_r and O_r are the resolutions of input and output to a layer while F, P and S indentify the filter/kernel, zero-padding and stride respectively.

$$O_r = \frac{(I_r - F + 2P)}{S} + 1 \tag{6.1}$$

$$O_r = S(I_r - 1) + F - 2P \tag{6.2}$$

Transfer learning [50] is an emerging trend in deep learning research where instead of training new network from scratch, existing networks are optimized and fine-tuned on one's own relevant data set. This is inspired from the fact that training these networks is a time consuming task if done from scratch and realization that the earlier layers in any deep network are more general. We have tested some of the FCNs for sky segmentation that were fine-tuned on different datasets by [42]. We compare segmentation results using different approaches i.e. how to obtain sky segmentation from the multiclass output

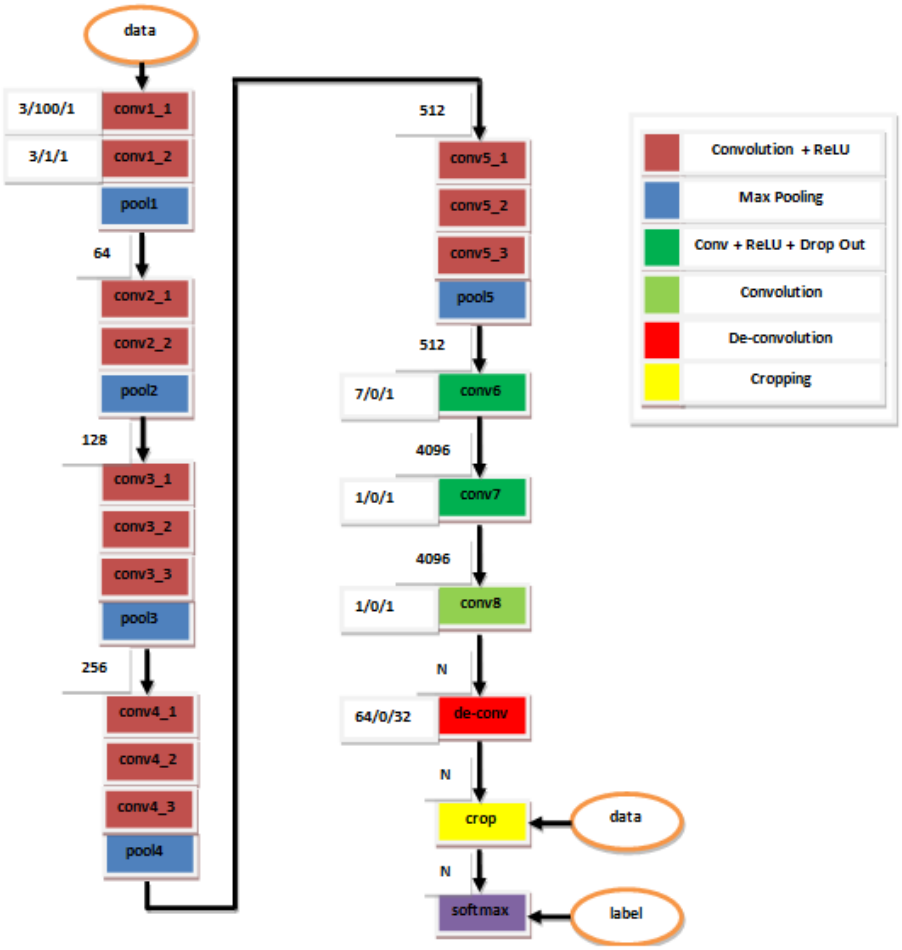

Figure 6.1: VGG[48] network transformed into FCN32s[42]

given by the networks into binary output (sky vs non-sky). Additionally, we have fine-tuned these existing networks on CH1 dataset to see if the existing multiclass segmentation network can be fine-tuned to perform better for the task of binary classification.

6.1.3 Deep Convolutional Encoder-Decoder Architecture : SegNet

SegNet [43] is also motivated by the same principles as FCN and is based on fully convolutional layers and does not involve any fully connected layer, however it further focuses on maintaining sharp boundary delineation which is essential for pixel-wise segmentation of small/rare classes. Additionally, unlike FCN which requires the stage-wise training where a new decoder (fusion at multiple strides) is progressively added to the existing trained network (FCN-32s) and the resulting network (FCN-16s) is trained again; SegNet provides the capability of end-to-end training, thanks to its decoder network. The encoder network in SegNet is exactly same as that of FCN i.e. 13 convolutional layers from VGG16 [48], however it is followed by a decoder containing equal number of convolutional layers as in encoder and is the core of SegNet. Each decoder in the decoder network first upsample its input coming from the corresponding encoder (max-pooling indices) in the encoder network and then followed by learnable convolutional layers. The final convolutional layer is followed by a classification layer (softmax) same as in FCN. This decoder-decoder architecture of SegNet allows end-to-end training and crisp boundary delineation while FCN would have to rely on additional components/architectures e.g. CRFs and RNNs to achieve the similar objective [52].

6.2 Training Details for Each Compared Method

6.2.1 Our Edge-less Approach (DCSI)

For this comparison, we trained an SVM classifier based on CH1 data set. For each of the training images (203 total), the positive (horizon) key points are chosen along the ground truth horizon line while equal number of negative (non-horizon) key points are chosen from edge locations randomly. We also make sure that negative key points are not in a close vicinity of the ground truth horizon. A 16×16 normalized intensity patch is used as a feature vector. The trained SVM classifier is used to generate a dense classification score map where a normalized score [0–1] for each pixel reflects its probability of horizon-ness. The dense image is taken as a dense graph and dynamic programming is applied to find a shortest path from left-most column (graph stage) to right-most column (graph stage) which conforms to a detected horizon boundary as detailed in chapter 3 Figure 6.2 shows some of the generated dense classification images (column 3) for several images from the extended test set being used for this comparison and the resulting segmentations for our edge-less method (column 4).

6.2.2 ALE

ALE [53] is an energy minimization-based semantic segmentation framework adopted for sky extraction by Saurer et al. [33]. Specifically, the energy is predicted by a pixel-wise classifier trained on contextual and superpixel feature representations. Multiple bag-of-words representations over the random set

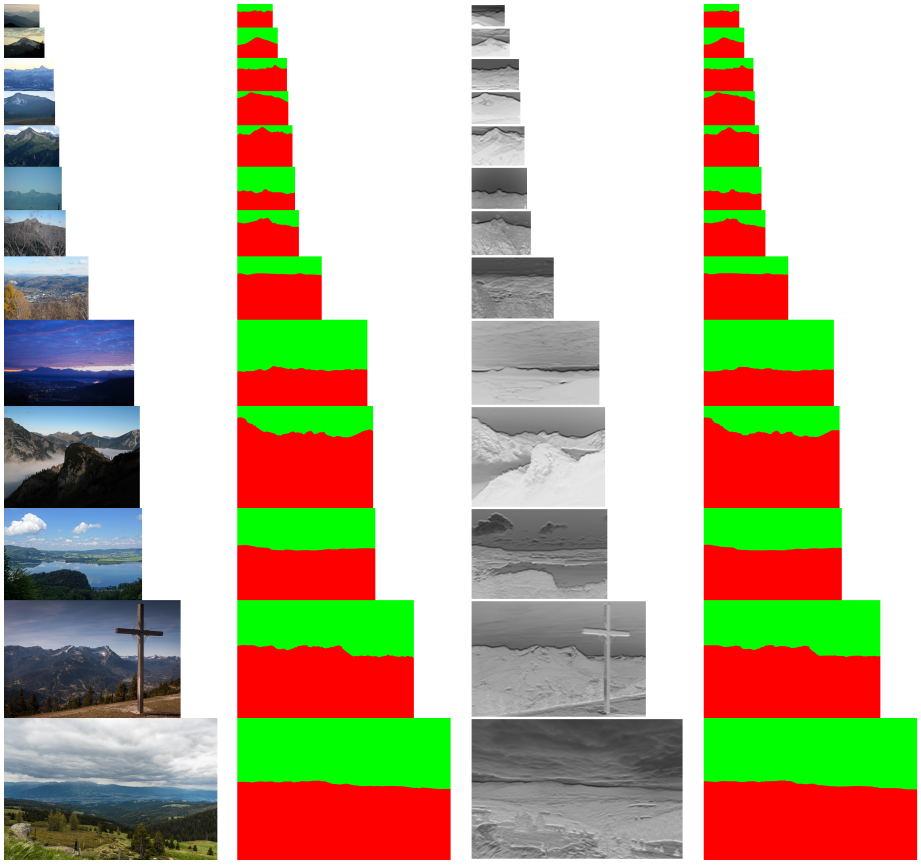

Figure 6.2: Edge-less skyline detection: query images (column 1) from the extended test set, ground truth (column 2), DCSIs (column 3) and corresponding segmentations (column 4).

of 200 rectangles, and superpixels are used for contextual part and superpixel part respectively. The segmentation is obtained by minimizing the energy using dynamic programming (DP). We implemented the algorithm [33] into the Automatic Labeling Environment (ALE) [53] environment. Similar to the original paper [33], we set the number of bag-of-words clusters to 512 and train ALE using CH1 dataset [17].

6.2.3 FCN

We used two different types of FCN models. PASCAL-context models were trained by the authors [42] on object and scene labeling of PASCAL VOC, in three different resolution capabilities (FCN32s, FCN16s and FCN8s with the highest resolution). The models include both object and surface classes (59 classes, including class "sky" and "mountain"). This type of network predicts scores for each class at each pixel location. SIFT Flow models were trained for joint geometric (3 classes: "sky", "horizontal", "vertical") and semantic (33 classes, including "sky" and "mountain") class segmentation and produces two separate scores. As all of the FCN models were trained for multiclass segmentation, we evaluated several methods based on how to compute binary segmentations (sky vs non-sky) from scores outputted by the networks. For PASCAL-context networks we compared scores for class 40 (mountain) and class 50 (sky) for each pixel. If the first score was higher the final class was set to "non-sky", otherwise it was set to "sky". For SIFT Flow networks there are more options, because the network provides two types of scores. We segmented a pixel as "sky" in case the highest semantic score was for class 28 ("sky"), otherwise it was segmented as "non-sky". Similarly, for the geometric score, we segmented a pixel as "sky" for case where the highest geometric score was for class 1 ("sky"), non-sky was segmented otherwise. For the best performing models we fine-tuned the weights of the pretrained networks. The network structure and training parameters were kept the same as for the original networks. The CH1 dataset was used to provide input images and binary labels (sky vs non-sky). For the

PASCAL-context pretrained network we modified binary labels so that the class indices correspond to correct classes in the network (class 40 denotes "mountain", class 50 denotes "sky"). For the SIFT Flow dataset, we had to convert source binary labels into two different target labels (geometric and semantic). Semantic label was set to class 28 ("sky") where the original label denoted sky, class 17 ("mountain") was used otherwise. Likewise, the geometric label was set to class 1 ("sky") where the original label denoted sky, class 3 ("vertical") was used otherwise. We expected such fine-tuned networks to perform better than the original pretrained networks. First reason is that the network is fine-trained with new unseen training data. The second is that the network is forced by the new input labels to predict only two classes ("mountain" and "sky"), so that such a new network is specialized for the task of sky and non-sky segmentation.

6.2.4 SegNet

Unfortunately, the publicly available models for SegNet were trained on urban images, specifically for semantic road scene segmentation. We have investigated different available models, and adopted the best performing SegNet model ("driving webdemo") for our sky segmentation problem. As mentioned earlier, SegNet is trained with urban imagery, using it directly for mountainous sky segmentation does not make sense. So, instead of using SegNet model directly, we first fine-tuned it with CH1 data set. Another disadvantage of SegNet is that both input image and output segmentation have fixed resolution (480x360), so we have to resize the input to this size

for training and later resized the 480x360 segmentations to the original sizes of respective images.

6.3 Experimental Details

6.3.1 Data Sets

For this comparison, our test set is comprised of 2895 mountainous images which have been acquired from Flickr. The GPS locations and camera intrinsic are used to access the relevant Digital Elevation Maps (DEMs) which are then rendered using a conventional OpenGL utility to develop the ground truth segmentations. These ground truth segmentations are used to compute the error metrics. For training edge-less classifier, ALE[33] and fine-tuning FCNs[42] and SegNet[43], we use the publicly available CH1 data set [17]. It should be noted that both training and test sets contain images with various resolutions compared to our earlier experiments where they have been converted to fix size. The deep-learning platform Caffe has been extensively used for training all the deep architectures – FCN and SegNet.

6.3.2 Error Metrics

In addition to use the average absolute pixel error defined in equation 4.1 as performance metric, we also compute the pixel level average accuracy obtained by each method. Specifically, for each of the images, we compute what fractions of pixels have been correctly classified by an approach. Mathemat-

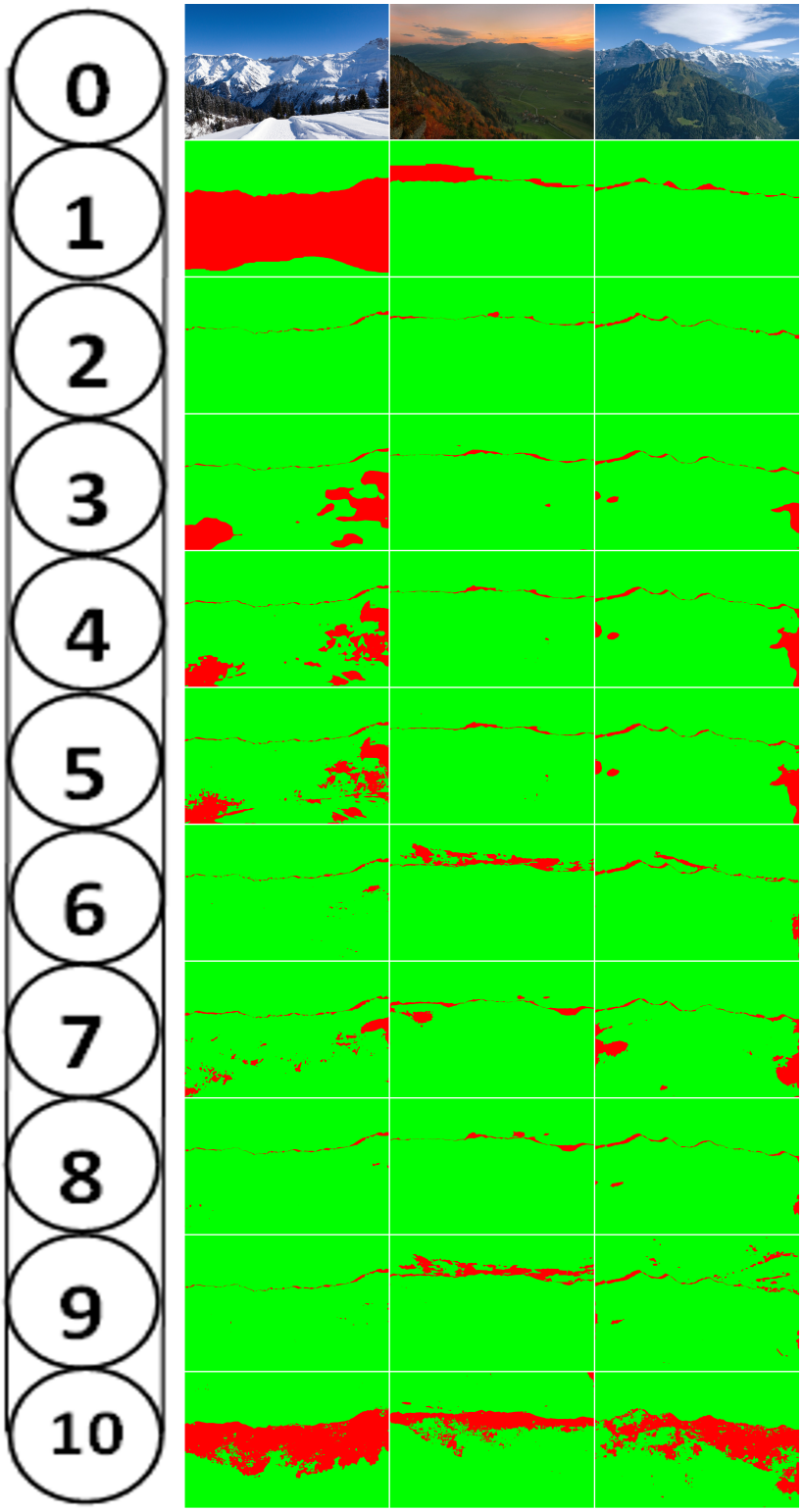

Figure 6.3: Some visual results for segmentation, green – correctly classified, red – miss-classified: (0) sample images from our data set, (1) DCSI, (2) ALE, (3) FCN32s-Pascal, (4) FCN16s-Pascal, (5) FCN8s-Pascal, (6) FCN8s-Pascal-CH1, (7) FCN8s-SiftFlow-geometric (8) FCN8s-SiftFlow-semantic, (9) FCN8s-SiftFlow-semantic-CH1 and (10) SegNet-CH1.

ically,

$$DC = \frac{1}{N_{set}} \sum_{i=1}^{N_{set}} \frac{{N_c}^i}{{N_t}^i} \tag{6.3}$$

where, N_{set} is the total number of images in the data set, ${N_c}^i$ and ${N_t}^i$ are the number of correctly classified pixels and number of total pixels in the image i. For an ideal 100% classification accuray this measure should result in a perfect 1.

6.3.3 Results

In the first set of experiments, we report the results for all the considered formulations without any post-processing. For DCSI and ALE; the classifiers are trained on CH1 data sets while different flavors of FCN models are either adopted as-it-is for sky segmentation or have been further trained through transfer learning on CH1 data set as noted in section 6.2.3. Table 6.1 lists both considered metrics for each of the formulations. The number after FCN i.e. (8s,16s or 32s) identifies the resolution at which the coarse semantic information has been fused with fine appearance information in the FCN architecture, SiftFlow or Pascal key-word identifies which data set has been used to train the model and an additional CH1 follows if the model has been fine-tuned further on CH1 data set. In case of SiftFlow, it is further distinguished between geometric (g) and semantic (s) models. Figure 6.3 shows some visual results.

Table 6.1: Performance of different formulations on our test set (2895 images).

Approach	Accuracy	Pixel Distance	
		μ	σ
FCN8s-Pascal	0.9083	29.886	50.721
FCN16s-Pascal	0.9071	30.187	50.616
FCN32s-Pascal	0.9015	31.160	50.714
FCN8s-SiftFlow-g	0.9266	37.028	56.481
FCN8s-SiftFlow-s	0.9438	37.937	67.608
Horizon-ALE-CH1	0.9428	44.669	87.430
Horizon-DCSI-CH1	0.8756	99.425	160.516
SegNet-CH1	0.8290	90.385	81.528
FCN8s-SiftFlow-s-CH1	0.9379	61.502	96.006
FCN8s-Pascal-CH1	0.9285	68.283	97.626

6.3.4 Post-Processing

Post-processing of the binary segmentation images can further improve the segmentation quality. As seen from examples in figure 6.3, some methods are able to find horizon accurately, but resulting segmentation images are not physically possible in reality. One example is to have large sky area surrounded by non-sky area ("hole in an object"), which is rare to happen in physical world. Another example is to have non-sky area surrounded by sky ("flying object"). Several post-processing methods that reflect physical world properties can be designed to improve quality of the resulting segmentation images. We adopted the following two simple post-processing approaches to further enhance the segmentation results.

It should be noted that not all the methods would benefit from such prost processing; specifically DCSI and ALE – as these methods employ Dynamic Programming to find crisp boundaries between sky and non-sky regions and

Table 6.2: Segmentation improvement due to Post-processing I

Approach	**Accuracy**	**Pixel Distance**	
		μ	σ
FCN8s-Pascal	0.9108	32.161	57.510
FCN16s-Pascal	0.9086	32.888	58.193
FCN32s-Pascal	0.9011	33.534	57.588
FCN8s-SiftFlow-g	0.9296	34.975	53.334
FCN8s-SiftFlow-s	0.9446	31.399	55.052
Horizon-ALE-CH1	0.9403	43.959	86.038
Horizon-DCSI-CH1	0.8727	99.742	160.252
SegNet-CH1	0.8279	114.893	99.021
FCN8s-SiftFlow-s-CH1	0.9421	37.947	69.435
FCN8s-Pascal-CH1	0.9351	41.596	71.707

mostly do not suffer due to miss-classification holes. Nonetheless, all the segmentations have been post-processed for consistency.

6.3.5 Post-processing I

The first post-processing method uses two basic binary image processing operations. The first operation fills all holes, sky areas that are fully surrounded by non-sky areas are replaced by non-sky area. The second operation removes small non-sky objects, specifically all small non-sky objects that have area below 50% of the largest non-sky object are removed, i.e. they are replaced by sky label. The post-processing improvements are listed in table 6.2.

6.3.6 Post-processing II

Similarly, two operations are used for the second post-processing method. The first operation removes small non-sky objects in the same way as in the first method. The second operation is column horizon detection, which finds

Table 6.3: Segmentation improvement due to Post-processing II

Approach	**Accuracy**	**Pixel Distance**	
		μ	σ
FCN8s-Pascal	0.9551	32.161	57.510
FCN16s-Pascal	0.9539	32.888	58.193
FCN32s-Pascal	0.9520	33.534	57.588
FCN8s-SiftFlow-g	0.9491	34.975	53.334
FCN8s-SiftFlow-s	0.9563	31.399	55.052
Horizon-ALE-CH1	0.9411	43.959	86.038
Horizon-DCSI-CH1	0.8743	99.742	160.252
SegNet-CH1	0.8437	114.893	99.021
FCN8s-SiftFlow-s-CH1	0.9486	37.947	69.435
FCN8s-Pascal-CH1	0.9432	41.596	71.707

first non-sky pixel label in a column from the top and sets all pixels below as non-sky pixel, i.e. first non-sky pixel in a column from the top defines the horizon. Table 6.3 shows the improvements due to this post-processing approach.

6.3.7 Discussion

While looking at the results reported in table 6.1 and considering only the accuracy, FCNs trained on SiftFlow clearly outperform the rest of the formulations. Surprisingly, the second best is the ALE approach which is a non-deep-learning method. This could be due to the fact that ALE is a well crafted approach for the problem of sky segmentation and may not generalize well to other segmentation applications unlike FCNs. The FCN8s – with the finest resolution outperforms the others with coarser fusion (FCN16s, FCN32) which follows the results reported by authors [42]. Interestingly, SegNet has performed abysmally, even worse than the simple SVM based

DCSI method. This is contradictory to the results reported by authors [43] for semantic road scene segmentation. It must be due to the fact that SegNet models made publicly available have been trained on urban street view data sets unlike FCNs. Another drawback of SegNet being the need to have images of fixed resolution which is not the case in FCN.

At first, the average absolute pixel error is not very consistent with the accuracy. Based on this measure alone, the poorly performing methods (i.e. DCSI and SegNet) can be readily identified and follow the observations made based on accuracy. However, from table 6.1 it follows that FCN8s-Pascal performs better than FCN8s-SiftFlow which is not the case considering accuracy alone. This interesting contradiction clears out in table 6.3 when FCN models based on Pascal benefit from post-processing II and all both FCNs have very close average pixel errors.

Post-processing has proven to be helpful for all the FCN formulations while it does not have much effect on dynamic programming based methods – DCSI and ALE. Although both post-processing methods impacted the segmentation results positively, the improvement due to second method of post-processing has been overall more effective.

Chapter 7

Visual Geo-Localization

As mentioned in earlier chapters; most of the available visual geo-localization techniques for mountainous imagery rely on users to correct for incorrect horizon segments – which is not always feasible. To address this, we propose a visual geo-localization pipeline which does not require an explicit detection of the skyline and utilizes dense classification score image resulting from our proposed edge-less/fusion approach in an evolutionary computing framework. Since, it has been previously established that orientation is more important than variation in the GPS [28, 17, 29, 31]; our focus is mostly on an accurate orientation estimate through geo-localization. Hence, in principle our approach is closer to that of [28]. In section 7.1, we discuss different components of our visual geo-localization framework. Section 7.2 lists the experimental details and results specific to geo-localization.

7.1 Proposed Pipeline

Figure 7.2 shows a block diagram of our visual geo-localization pipeline which is comprised of three main modules: 1) classifier training, 2) score map generation, and 3) particle swarm optimization and rendering. Details regarding each of these components are provided next.

7.1.1 DCSI as Fitness Function

We have noticed in chapter 3; classifying the pixels results into narrow band of high horizon-ness scores around the ground truth horizon. In our framework, we exploit these high scores around a true horizon to guide Particle Swarm Optimization (PSO) such that the synthetically generated horizon aligns well with ground truth horizon in the image. First, a synthetic horizon is rendered for a given orientation estimate and provided GPS location and camera intrinsic. The resultant rendered horizon is overlaid on the classification map for the respective query image. The horizon-ness scores or classification values for all the pixels of the rendered horizon are averaged and this average is used as a fitness score. As more and more solutions are explored; PSO guides itself towards the narrow band around true horizon where the fitness score is minimized. Mathematically, the fitness score $F(r)$ for a rendered candidate horizon r can be defined as,

$$F(r) = \frac{1}{N} \sum_{k=1}^{N} S(i_r(k), j_r(k)) \tag{7.1}$$

where, $S(i_r(k), j_r(k))$ is the classification score for k-th pixel of the ren-

dered horizon and follows from 3.12, and N is the number of pixels in the rendered horizon.

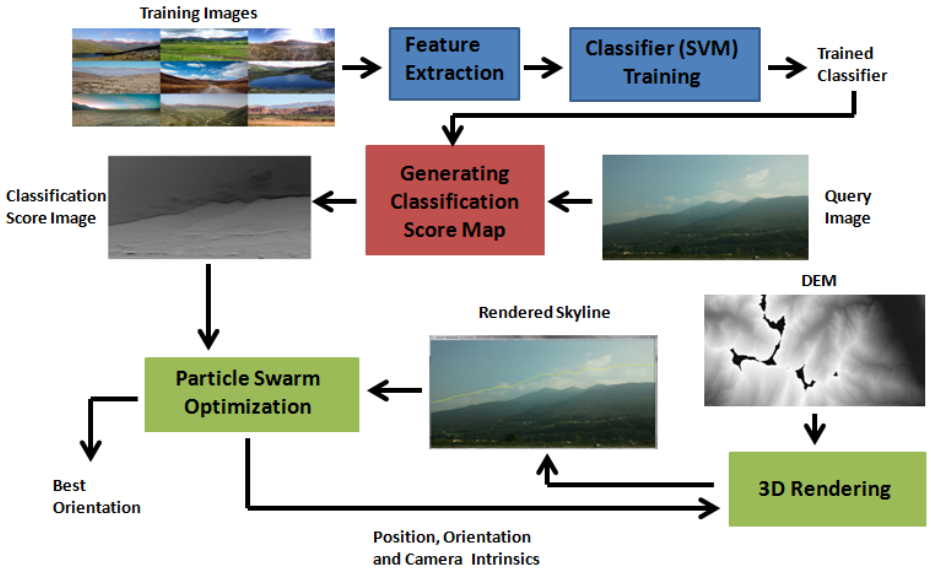

Figure 7.1: Proposed framework: Blue – feature extraction and classifier training performed offline, Red – module to generate classification score map for given image using trained classifier and Green – localization module comprised of PSO and rendering pipeline responsible for generating new solutions and rendering of synthetic skylines.

7.1.2 Particle Swarm Optimization (PSO)

We structure our visual geo-localization method in an evolutionary framework. Specifically, we rely on PSO which is an evolutionary algorithm that first generates random solutions to a given problem within a defined range and then converges to one of the best solutions based on the provided fitness function. For localization problem, since, the rough estimate of the orientation is mostly available from device sensors a neighborhood search window

around such a rough estimate is first defined. The PSO then generates random estimates for each of the defined parameter and guides itself to good solutions based on the defined fitness score. We utilize our DCSI as fitness according to 7.1 for guiding PSO towards a best alignment between synthetic and imaged horizon without even requiring explicit detection of the imaged horizon.

7.1.3 Rendering the Synthetic Skyline

We adopt a conventional graphics rendering pipeline to generate synthetic skylines from the DEM. Since, the GPS co-ordinates are already known, the search is constrained to a restricted patch of 3D model which is partially motivated due to computational requirements to load and render a big 3D map. The rendering pipeline can be considered as a function of three variables – 1) 3D DEM patch P, 2) camera intrinsic matrix K and 3) orientation of the camera expressed as a three vector $X = [u, v, w]$, where u, v and w are yaw, pitch and roll angles respectively. Mathematically, the rendering is a projection from 3D profile to 2D curve,

$$r = Proj(P, K, X) \tag{7.2}$$

where, r is the rendered horizon projected in 2D. The pixel locations for this synthetic skyline are then used for calculating the fitness score as described in section 7.1.1. Figure shows a rendered horizon line which is then overlaid on corresponding DCSI image that is being used as fitness function to guide PSO.

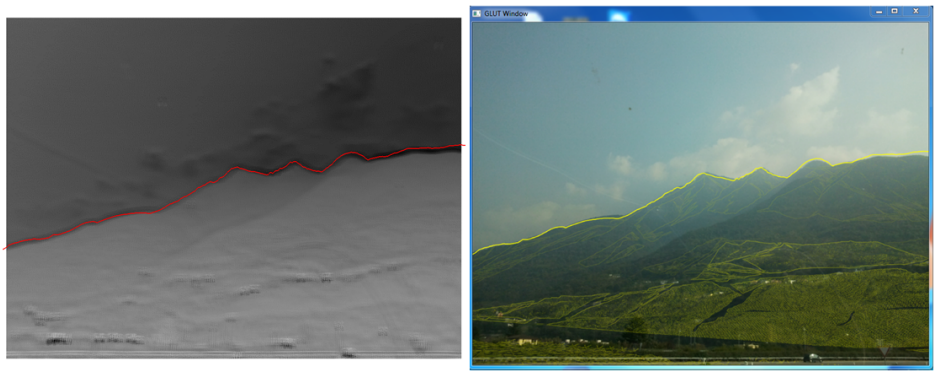

Figure 7.2: Renderred synthetic horizon for a sample orientation (left), same projected horizon overlaid on the corresponding DCSI image (right).

7.2 Experimental Details and Results

7.2.1 Data Set

We have used samples from CH1 data set as our test set for visual geo-localization experiments, whereas selected images from CH2 data set are being used to train SVM classifier. These images have been taken across Switzerland and provide the required terrain and seasonal variations. In addition of ground truth segmentations, authors [17] provide the ground truth GPS location and estimate of the camera focal length. However, the ground truth orientation estimates have not been provided. In a first step towards our goal, we manually aligned the imaged horizon with synthetic horizon rendered from the corresponding DEMs. This provides the ground truth orientation estimate which serves two purposes: (i) to define a search window around the known orientation estimate and (ii) to quantify how close the PSO based orientation estimates are with the ground truths.

Figure 7.3 shows the GPS locations marked on a map where CH1 data set

Figure 7.3: GPS locations for CH1 data set images marked on a map.

images have been taken in Switzerland. Multiple images have been taken at the same locations but with varied orientations and hence covering different mountains/terrains. The DEMs covering these image locations have been shown in figure 7.4 overlaid on a map in Google Earth Pro. This is worth mentioning that a mountain viewed in an image might not be covered by a single DEM slice (highlighted by white rectangles in figure 7.4) and required stitching of multiple DEM slices.

7.2.2 Quantitative Measures and Results

To quantitatively evaluate the performance of our proposed visual geo-localization pipeline, we have adopted two measure – 1) average pixelwise absolute distance between the PSO's best solution and ground truth skyline and 2) average error between the ground truth and estimated orientation for roll, yaw

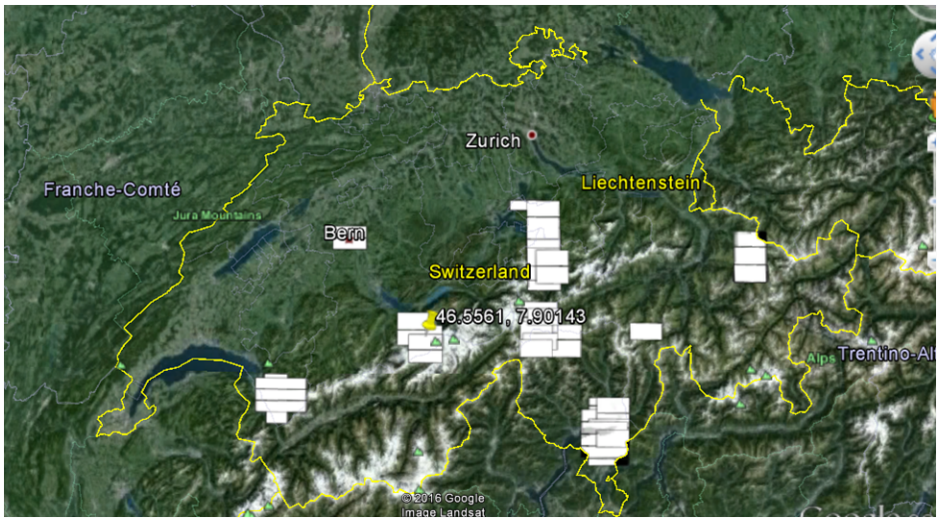

Figure 7.4: The DEMs surrounding the image locations overlaid in Google Earth Pro.

and pitch angles measured in degrees. Mathematically, the average absolute pixel error is the same as adopted for measuring the accuracy of skyline detection in chapter 4.

$$S = \frac{1}{N} \sum_{j=1}^{N} |P_{s(j)} - P_{g(j)}| \tag{7.3}$$

where $P_{s(j)}$ and $P_{g(j)}$ are the positions (rows) of the PSO's solution and ground horizon pixels in column j respectively and N is the number of columns in the given image.

Table 7.1: Absolute vertical distance between ground truth and projected solution skylines.

Vertical Pixel Distance	
Mean	Std
10.27	23.91

Table 7.2: Alignement error for orientation angles in degrees

X-angle		**Y-angle**		**Z-angle**	
Mean	Std	Mean	Std	Mean	Std
1.43	3.19	3.32	8.09	0.86	1.47

7.2.3 Discussion

As clear from table 7.1, the average absolute pixel error between ground truth horizons and the rendered horizons is quite high. This high average is due to the fact that for some images the rendered horizon does not align with the pictured horizon. There could be two reasons for misalignment: 1) The found solution is aligned such that it minimizes the fitness score but the pixels responsible for producing best classification scores are not in the vicinity of true horizon. In essence, this boils down to poor classification accuracy as there are curve segments which produce lower classification scores than the actual horizon. 2) The DEM being used is missing portion of the mountain which has been pictured in the photograph. Instances of this problem can be seen in figure 7.5 where segments of the found synthetic solution are way far from the true horizon. In our error metric, as the absolute error is computed regardless of problematic DEMs, the overall average error becomes high.

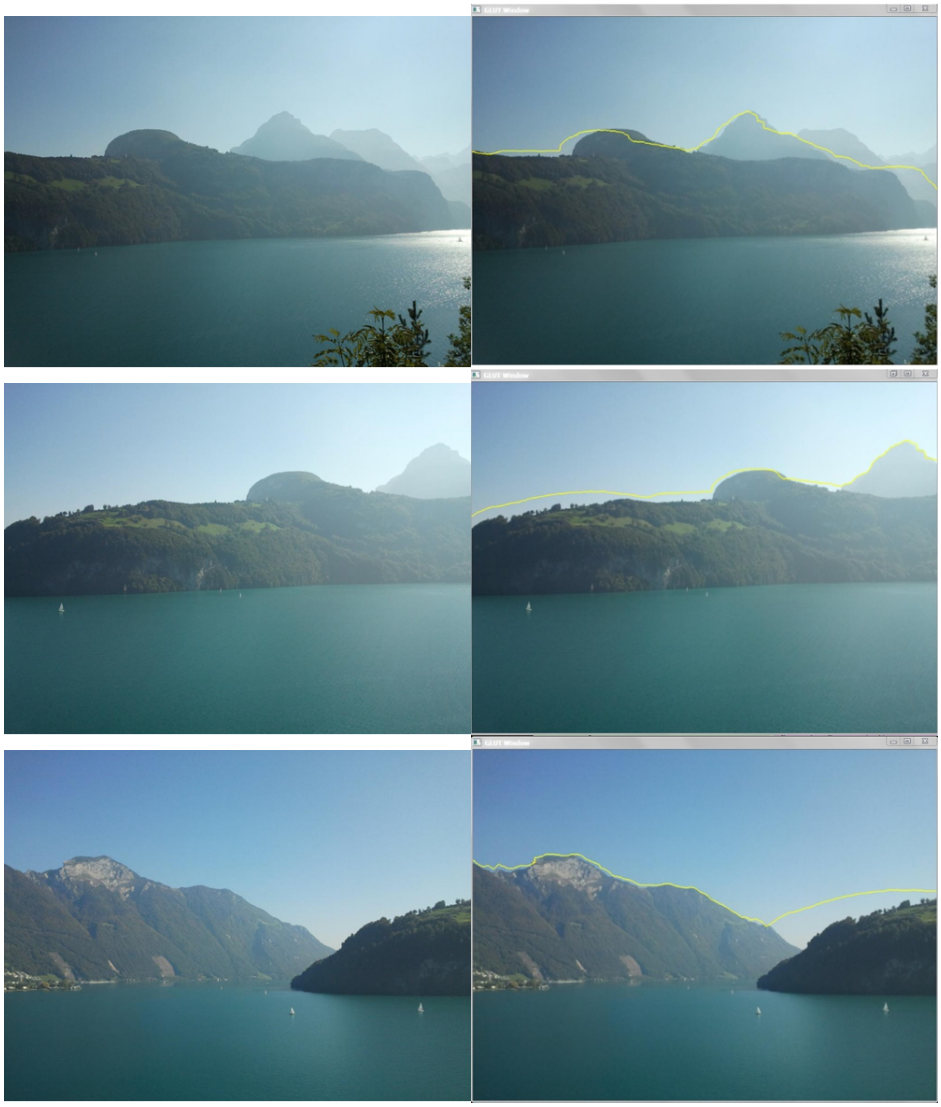

Figure 7.5: Sample images where synthetically renderred horizon does not align with true horizon due to incomplete DEM: (column1) query image and (column2) respective best renderred solution.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In this thesis, we have proposed two novel horizon detection approaches based on supervised machine learning. The first approach relies on edge detection but unlike conventional methods; it uses trained classifier to reduce the number of horizon candidate edges considerably. Further, classifier is used to provide evidence about the likelihood of an edge being horizon or non-horizon. We have investigated various texture features and nodal costs in the framework of our proposed method. Our second proposed approach does not rely on edge detection as a pre-processing step. In the edge-less approach, the key idea is to generate a classification score map and to apply DP on the resultant graph to extract the horizon line.

Both of the proposed approaches do not make any assumptions about the horizon being a straight line or being close to the top of the image which is a common assumptions in conventional edge based methods. The proposed

approaches use a very small number of images to train the horizon classifiers and outperform traditional approaches based on edge maps on two data sets. The underlying failure reasons for both approaches are investigated and a fusion is proposed which has been evaluated on an additional data set. The formulation of edge-less/fusion method has been shown to be useful for solution verification and detection of absence of horizon and partial horizon detection.

Impressed with the recent success of deep learning architectures, we adopted recently proposed two such networks for sky segmentation problem and compared them against our edge-less approach on an even larger test set. For a fair comparison, we have also included another prominent skyline detection approach which has been recently proposed and is an instance of classical feature based classifier training. Lastly, we have proposed a visual geo-localization pipeline which unlike previous solutions does not rely on human users and does not require an accurately detected horizon. This results from our edge-less skyline detection formulation and based on the observation that higher horizon-ness confidence is achieved by the pixels around the true horizon.

8.2 Potential Future Work

While we have formulated skyline detection problem as an instance of deep learning, the networks are fine-tuned on rather small data sets. In a future study, bigger data set can be used for training various deep networks and a quantitative comparison can be provided. For visual geo-localization, only

SVM classifier has been used to generate DCSI, different classifiers including deep networks can be adopted to study their role. As, we have noted in chapter 4, CNN tends to generate crisp boundaries compared to SVM, that might be more helpful for guiding PSO in less iterations. Additionally, geo-localization based on partial matching of synthetic and pictured horizon can be explored. As described in section 7.2.3, incomplete DEM could result in absolute average error to become higher, a more controlled metric can be adapted where only those segments of synthetic horizon are considered which have proper 3D model.

Publications

A list of the publications which are included in this dissertation.

Journal Publications

3. **T. Ahmad**, E. Emami, G. Bebis, "Horizon and Skyline Detection Methods and their Applications: A Review", (**Writing**)

2. **T. Ahmad**, G. Bebis, M. Nicolescu, A. Nefian and T. Fong, "Horizon Line Detection: Edge-less vs Edge-based Approaches and their Fusion", **Computer Vision and Image Understanding**, 2017. (**Revisions Sumbitted**)

1. **T. Ahmad**, G. Bebis, E. Regentova, A. Nefian and T. Fong, "Coupling Dynamic Programming with Machine Learning for Horizon Line Detection", **International Journal on Artificial Intelligence Tools**, Vol. 24, No. 4, 2015.

Conference Publications

6. **T. Ahmad**, E. Emami, G. Bebis, "Towards Mountainous Visual Geo-Localization without Explicit Sky Line Detection", (**Writing**)

5. **T. Ahmad**, P. Campr, M. Cadik and G. Bebis, "Comparison of Semantic Segmentation Approaches for Horizon/Sky Line Detection", **IEEE International Joint Conference on Neural Networks (IJCNN'17)**, Anchorage, Alaska, USA, May 14-19, 2017.

4. **T. Ahmad**, G. Bebis, M. Nicolescu, A. Nefian and T. Fong, "An Edge-less Approach to Horizon Line Detection", **14th IEEE International Conference on Machine Learning and Applications (ICMLA'15)**, Miami, Florida, USA. Dec 9-11, 2015.

3. **T. Ahmad**, G. Bebis, M. Nicolescu, A. Nefian and T. Fong, "Fusion of Edge-less and Edge-based Approaches for Horizon Line Detection", **6th IEEE International Conference on Information, Intelligence, Systems and Applications (IISA'15)**, Corfu, Greece, July 6-8, 2015.

2. **T. Ahmad**, G. Bebis, E. Regentova, A. Nefian and T. Fong, "An Experimental Evaluation of Different Features and Nodal Costs for Horizon Line Detection", **10th International Symposium on Visual Computing (ISVC'14)**, Las Vegas, Nevada, USA, Dec 8-10, 2014.

1. **T. Ahmad**, G. Bebis, E. Regentova and A. Nefian, "A Machine Learning Approach to Horizon Line Detection Using Local Features.", **9th International Symposium on Visual Computing (ISVC'13)**, Crete, Greece, July 29-31, 2013.

Publications which are not part of this dissertation, but are authored during Ph.D.

Journal Publications

1. Y. Chu, **T. Ahmad**, G. Bebis and L. Zhao, "Low-resolution Face Recognition with Single Sample per Person,", **Signal Processing**, Vol. 141, pp. 144-157, 2017.

Conference Publications

3. **T. Ahmad**, D. Ilstrup, E. Emami and G. Bebis, "Symbolic Road Marking Recognition Using Convolutional Neural Networks", **28th IEEE Intelligent Vehicles Symposium (IV'17)**, Redondo Beach, California, USA. June 11-14, 2017.

2. M. Saadi, **T. Ahmad**, Y. Zhao and L. Wuttisttikulkij, "An LED based Indoor Localization System using k-means Clustering", **15th IEEE International Conference on Machine Learning and Applications (ICMLA'16)**, Anaheim, California, USA. Dec 18-20, 2016.

1. A. P. Yazdanpanah, E. Regentova, A. K. Mandava, **T. Ahmad** and G. Bebis, "Sky Segmentation by Fusing Clustering with Neural Networks.", **9th International Symposium on Visual Computing (ISVC'13)**, Crete, Greece, July 29-31, 2013.

Bibliography

[1] T. Ahmad, G. Bebis, E. Regentova and A. Nefian, A Machine Learning Approach to Horizon Line Detection using Local Features, *Proceedings of 9th International Symposium on Visual Computing (ISVC)*. 2013.

[2] Y. Hung, C. Su, Y. Chang, J. Chang and H. Tyan, Skyline Localization for Mountain Images, *Proceedings of International Conference on Multimedia and Expo (ICME)*. 2013.

[3] W. Lie, T. C.-I. Lin , T. Lin , and K.-S. Hung, A robust dynamic programming algorithm to extract skyline in images for navigation, in *Pattern Recognition Letters*, **26**(2)(2005.)221–230.

[4] N. S. Boroujeni, S. A. Etemad and A. Whitehead: Robust Horizon Detection Using Segmentation for UAV Applications. *Proceedings of IEEE 2012 Ninth Conference on Computer and Robot Vision*, 2012.

[5] G. C. H. E. de Croon, B. D. W. Remes, C. De Wagter and R. Ruijsink: Sky Segmentation Approach to Obstacle Avoidance. *IEEE Aerospace Conference*, 2011.

[6] S. M. Ettinger, M. C. Nechyba, P. G. Ifju and M. Waszak: Vision-Guided Flight Stability and Control for Micro Air Vehicles. *Proceedings of International Conference on Intelligent Robots and Systems(IEEE/RSJ)*, 2002.

[7] T. G. McGee, R. Sengupta and K. Hedrick: Obstacle Detection for Small Autonomous Aircraft Using Sky Segmentation. *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2005.

[8] S. Thurrowgood, D. Soccol, R. J. D. Moore, D. Bland and M. V. Srinivasan: A Vision Based System for Altitude Estimation of UAVs. *Proceedings of International Conference on Intelligent Robots and Systems(IEEE/RSJ)*, 2009.

[9] S. Todorovic, M. C. Nechyba and P. G. Ifju: Sky/Ground Modeling for Autonomous MAV Flight. *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2003.

[10] V. Gupta and S. Brennan : Terrain Based Vehicle Orientation Estimation Combining Vision and Inertial Measurements. *Journal of Field Robotics*, **25**(3):181 - 202, 2008.

[11] N. Ho and P. Chakravarty: Localization on Freeways using the Horizon Line Signature. *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2014.

[12] S. J. Dumble and P. W. Gibbens: Efficient Terrain-Aided Visual Horizon Based Attitude Estimation and Localization. Journal of Intelligent & Robotic Systems, **78**(2):205–221, (2015)

[13] T. Ojala, M. Pietikainen and T. Maenpaa: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence(TPAMI)*, **24**(7):971 - 987, 2002.

[14] D. G. Lowe: Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision(IJCV)*, **68**(2):91 - 110, 2004.

[15] N. Dalal and B. Triggs: Histograms of Oriented Gradients for Human Detection. *Proceedings of Computer Vision and Pattern Recognition(CVPR)*, 2005.

[16] http://www.vlfeat.org/index.html

[17] G. Baatz, O. Saurer, K. Koser and M. Pollefeys: Large Scale Visual Geo-Localization of Images in Mountainous Terrain *ECCV*, 2012.

[18] W. Liu and C. Su: Automatic Peak Recognition for Mountain Images *Advanced Technologies, Embedded and Multimedia for Human-centric Computing*, 2014.

[19] S. Fefilatyev, V. Smarodzinava, L. O. Hall and D. B. Goldgof: Horizon Detection Using Machine Learning Techniques. *ICMLA.*, 17-21, 2006.

[20] E. Gershikov, T. Libe and S. Kosolapov: Horizon Line Detection in Marine Images: Which Method to Choose? *International Journal on Advances in Intelligent Systems*, **6**(1-2):79 - 88, 2013.

[21] B.-J. Kim, J.-J. Shin, H.-J. Nam and J.-S. Kim: Skyline Extraction using a Multistage Edge Filtering *World Academy of Science, Engineering and Technology 55*, 2011.

[22] A. P. Yazdanpanah, E. E. Regentova, A. K. Mandava, T. Ahmad and G. Bebis: Sky Segmentation by Fusing Clustering with Neural Networks. *Proceedings of 9th International Symposium on Visual Computing (ISVC)*. 2013.

[23] A. V. Nefian, X. Bouyssounouse, L. Edwards, T. Kim, E. Hand, J. Rhizor, M. Deans, G. Bebis and T. Fong: Planetary Rover Localization within Orbital Maps. *Proceedings of International Conference on Image Processing(ICIP)*. 2014.

[24] T. Ahmad, G. Bebis, E. Regentova, A. Nefian and T.Fong: An Experimental Evaluation of Different Features and Nodal Costs for Horizon Line Detection, *Proceedings of 10th International Symposium on Visual Computing (ISVC)*. 2014.

[25] J. Matas, O. Chum, M. Urban, and T. Pajdla: Robust Wide Baseline Stereo from Maximally Stable Extremal Regions, *Proceedings of British Machine Vision Conference*, pages 384-396, 2002.

[26] C. Cortes and V. Vapnik: Support-vector networks., *Machine Learning*, **20**(3):273 - 297, 1995.

[27] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient based learning applied to document recognition., *PIEEE*, **86**(11):2278–2324, 1998.

[28] L. Baboud, M. Cadik, E. Eisemann and H. -P Seidel: Automatic Photo-to-Terrain Alignment for the Annotation of Mountain Pictures. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[29] E. Tzeng, A. Zhai, M. Clements, R. Townshend and A. Zakhor: User-Driven Geolocation of Untagged Desert Imagery Using Digital Elevation Models. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops(CVPRW)*, 2013.

[30] J. Canny: A computational approach to edge detection., *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **8**(6):679–698, 1986.

[31] L. Porzi, S. R. Bulo, P. Valigi, O. Lanz and E. Ricci: Learning Contours for Automatic Annotations of Mountains Pictures on a Smartphone., *ACM/IEEE Int. Conf. on Distributed Smart Cameras*, 2014.

[32] Y. Chen, G. Qian, K. Gunda, H. Gupta and K. Shafique: Camera Geolocation From Mountain Images, *IEEE Int. Conf. on Information Fusion*, 2015.

[33] O. Saurer, G. Baatz, K. Koser, L. Ladicky and M. Pollefeys: Image Based Geo-localization in the Alps, *International Journal of Computer Vision (IJCV)*, **116**(3):213–225, 2016.

[34] R. Verbickas and A. Whitehead: Sky and Ground Detection Using Convolutional Neural Networks. *International Conference on Machine Vision and Machine Learning (MVML)*, 2014.

[35] A. M. Neto, A. C. Victorino, I. Fantoni and D. E. Zampieri: Robust Horizon Finding Algorithm for Real-Time Autonomous Navigation based on Monocular Vision. *International Conference on Intelligent Transportation Systems*, 2011.

[36] C. Liu, Y. Zhang, K. Tan and H. Yang: Sensor Fusion Method for Horizon Detection From an Aircraft in Low Visibility Conditions. *IEEE Transactions on Instrumentation and Measurement*, **63**(3):620–627, 2014.

[37] Y. Shen and Q. Wang: Sky Region Detection in a Single Image for Autonomous Ground Robot Navigation. *International Journal of Advanced Robotic Systems*, **10**, 2013.

[38] Y. Shen, D. Krusienski, J. Li and Z. Rahman: A Hierarchical Horizon Detection Algorithm. *IEEE Geoscience and Remote Sensing Letters*, **10**(1):111–114, 2013.

[39] D. Dusha, W. Boles and R. Walker: Attitude Estimation for a Fixed-Wing Aircraft Using Horizon Detection and Optical Flow. *Proceedings of Digital Image Computing Techniques and Applications (DICTA)*, pp. 485–492, 2007.

[40] E. Gershikov: Is Color Important for Horizon Line Detection? *Proceedings of International Conference on Advanced Technologies for Communications*, pp. 262–267, 2014.

[41] D. Braun and M. Singhof: Automated Silhouette Extraction for Mountain Recognition. *Proceedings of 27th GI-Workshop*, pp. 18–23, 2015.

[42] J. Long, E. Shelhamer and T. Darrell: Fully Convolutional Networks for Semantic Segmentation *CVPR*, 2015.

[43] V. Badrinarayanan, A. Kendall and R. Cipolla: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation *arXiv:1511.00561v2 [cs.CV]*, 2015.

[44] J. Hays and A. A. Efros: IM2GPS: estimating geographic information from a single image *Computer Vision and Pattern Recognition (CVPR)*, 2008.

[45] Y. Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, T. Chua, H. Neven and J. Yagnik: Tour the world: building a web-scale landmark recognition engine *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[46] A. R. Zamir and M. Shah: Accurate image localization based on Google maps street view. *Proceedings of European Conference on Computer Vision (ECCV)*, 2010.

[47] A. Krizhevsky, I. Sutskever and G. E. Hinton: Imagenet Classification with Deep Convolutional Neural Networks *NIPS*, 2012.

[48] K. Simonyan and A. Zisserman: Very Deep Convolutional Networks for Large-scale Image Recognition *CoRR, abs/1409.1556*, 2014.

[49] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich: Very Deep Convolutional Networks for Large-scale Image Recognition *CoRR, abs/1409.4842*, 2014.

[50] J. Yosinski, J. Clune, Y. Bengio and H. Lipson: How transferable are features in deep neural networks? *NIPS*, 2014.

[51] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov: Dropout: A Simple Way to Prevent Neural Networks from Overfitting *The Journal of Machine Learning Research (JMLR)*, **15**(1):1929–1958, 2014.

[52] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang and P. H. S. Torr: Conditional Random Fields as Recurrent Neural Networks *arXiv:1502.03240[cs.CV]*, 2015.

[53] L. Ladicky, C. Russell, P. Kohli and P. H. S. Torr: Graph Cut Based Inference with Co-occurrence Statistics *Proceedings of the 11th European Conference on Computer Vision (ECCV)*, 2010.

[54] B. Grelsson, M. Felsberg and F. Isaksson: Highly Accurate Attitude Estimation via Horizon Detection *Journal of Field Robotics*, 2015.

[55] J. Hou and B. Li: An Improved Algorithm for Horizon Detection Based on OSTU *Proceedings of International Conference on Intelligent Human-Machine Systems and Cybernetics*, 2015.

[56] L. Di, T. Fromm and Y. Chen: A Data Fusion System for Attitude Estimation of Low-cost Miniature UAVs *Journal of Intelligent & Robotic Systems*, **65**(1):621–635, 2012.

[57] E. Boukas, A. Gasteratos and G. Visentin: Localization of Planetary Exploration Rovers with Orbital Imaging: a survey of approaches *Inter-*

national Conference on Robotics and Automation Workshops (ICRAW), 2014.

[58] F. Cozman and E. Krotkov: Automatic Mountain Detection and Pose Estimation for Teleoperation of Lunar Rovers *Proceedings of International Conference on Robotics and Automation (ICRA)*, 1997.

[59] F. Cozman, E. Krotkov and C. Guestrin: Outdoor Visual Position Estimation for Planetary Rovers *Autonomous Robots*, **9**(2):135–150, 2002.

[60] W. Kruger and Z. Orlov: Robust Layer-based Boat Detection and Multi-target-tracking in Maritime Environments *Proceedings of International Waterside Security Conference*, 2010.

[61] X. Kong, L. Liu, Y. Qian and M. Cui: Automatic detection of sea-sky horizon line and small targets in maritime infrared imagery *Infrared Physics & Technology*, **76**:185–199, 2016.

[62] A. P. Yazdanpanah, E. E. Regentova, V. Muthukumar and G. Bebis: Efficient Terrain-Aided Visual Horizon Based Attitude Estimation and Localization *International Journal of Computer Applications*, **121**(10), 2015.

[63] F. Stein and G. Medioni: Map-based Localization Using the Panoramic Horizon *IEEE Transactions on Robotics and Automation*, **11**(6):892–896, 1995.

[64] P. Chippendale, M. Zanin and C. Andreatta: Spatial and Temporal Attractiveness Analysis Through Geo-Referenced Photo Alignment *Pro-*

ceedings of IEEE International Geoscience and Remote Sensing Symposium, 2008.

[65] O. C. Ozcanli, Y. Dong and J. L. Mundy: Geo-localization using Volumetric Representations of Overhead Imagery *International Journal of Computer Vision*, **116**(3):226–246, 2016.

[66] R. Fedorov and P. Fraternali and M. Tagliasacchi: Mountain Peak Identification in Visual Content Based on Coarse Digital Elevation Models *Proceedings of 3rd International Workshop on Multimedia Analysis for Ecological Data*, 2014.

[67] L. Nicolle, J. Bonneton, H. Konik, D. Muselet, and L. Tougne: Towards an electronic orientation table: using features extracted from the image to register Digital Elevation Model. *Int. Conf. on Computer Vision Theory and Applications (VISAPP)*, 2017.

[68] T. Ahmad, G. Bebis, M.Nicolescu, A. Nefian, and T. Fong: An Edge-Less Approach to Horizon Line Detection. *Int. Conf. on Machine Learning and Applications*, 2015.

[69] L. Porzi, S. R. Bulo, O. Lanz, P. Valigi, and E. Ricci: Image Based Geo-localization in the Alps. *Machine Vision and Applications (MVA)*, **28**(1-2):101–115, 2017.